Libraries Guide
About This Guide

The Libraries Guide is part of the ISE documentation collection.

Guide Contents

This guide contains the following:

- Discussion of the Xilinx Unified Libraries
- Slice Count information for FPGAs
- A listing of the various Functional Categories of design elements
- Architecture Specific Information chapters
- Individual chapters for each of the Design Elements

Additional Resources

For additional information, go to http://support.xilinx.com. The following table lists some of the resources you can access from this website. You can also directly access these resources using the provided URLs.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutorials</td>
<td>Tutorials covering Xilinx design flows, from design entry to verification and debugging</td>
</tr>
<tr>
<td></td>
<td><a href="http://support.xilinx.com/support/techsup/tutorials/index.htm">http://support.xilinx.com/support/techsup/tutorials/index.htm</a></td>
</tr>
<tr>
<td>Answer Browser</td>
<td>Database of Xilinx solution records</td>
</tr>
<tr>
<td></td>
<td><a href="http://support.xilinx.com/xlnx/xil_ans_browser.jsp">http://support.xilinx.com/xlnx/xil_ans_browser.jsp</a></td>
</tr>
<tr>
<td>Application Notes</td>
<td>Descriptions of device-specific design techniques and approaches</td>
</tr>
<tr>
<td></td>
<td><a href="http://support.xilinx.com/apps/appsw">http://support.xilinx.com/apps/appsw</a> eb.htm</td>
</tr>
<tr>
<td>Data Book</td>
<td>Pages from The Programmable Logic Data Book, which contains device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging</td>
</tr>
<tr>
<td></td>
<td><a href="http://support.xilinx.com/partinfo/databook.htm">http://support.xilinx.com/partinfo/databook.htm</a></td>
</tr>
<tr>
<td>Problem Solvers</td>
<td>Interactive tools that allow you to troubleshoot your design issues</td>
</tr>
<tr>
<td></td>
<td><a href="http://support.xilinx.com/support/troubleshoot/psolvers.htm">http://support.xilinx.com/support/troubleshoot/psolvers.htm</a></td>
</tr>
<tr>
<td>Tech Tips</td>
<td>Latest news, design tips, and patch information for the Xilinx design environment</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.support.xilinx.com/xlnx/xil_tt_home.jsp">http://www.support.xilinx.com/xlnx/xil_tt_home.jsp</a></td>
</tr>
</tbody>
</table>
Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning or Use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier font</td>
<td>Messages, prompts, and program files that the system displays</td>
<td>speed grade: -100</td>
</tr>
<tr>
<td>Courier <strong>bold</strong></td>
<td>Literal commands that you enter in a syntactical statement</td>
<td><code>ngdbuild design_name</code></td>
</tr>
<tr>
<td><strong>Helvetica bold</strong></td>
<td>Commands that you select from a menu</td>
<td><code>File → Open</code></td>
</tr>
<tr>
<td></td>
<td>Keyboard shortcuts</td>
<td><code>Ctrl+C</code></td>
</tr>
<tr>
<td><em>Italic font</em></td>
<td>Variables in a syntax statement for which you must supply values</td>
<td><code>ngdbuild design_name</code></td>
</tr>
<tr>
<td></td>
<td>References to other manuals</td>
<td>See the Development System Reference Guide for more information.</td>
</tr>
<tr>
<td></td>
<td>Emphasis in text</td>
<td>If a wire is drawn so that it overlaps the pin of a symbol, the two nets are not connected.</td>
</tr>
<tr>
<td>Square brackets</td>
<td>An optional entry or parameter. However, in bus specifications, such as <strong>bus[7:0]</strong>, they are required.</td>
<td><code>ngdbuild [option_name] design_name</code></td>
</tr>
<tr>
<td></td>
<td>Braces { }</td>
<td>`lowpwr = {on</td>
</tr>
<tr>
<td></td>
<td>Vertical bar</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vertical ellipsis . . .</td>
<td>IOB #1: Name = QOUT'</td>
</tr>
<tr>
<td></td>
<td>Horizontal ellipsis ...</td>
<td>IOB #2: Name = CLKIN'</td>
</tr>
</tbody>
</table>

Online Document

The following conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning or Use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blue text</strong></td>
<td>Cross-reference link to a location in the current document</td>
<td>See the section “Additional Resources” for details.</td>
</tr>
<tr>
<td><strong>Red text</strong></td>
<td>Cross-reference link to a location in another document</td>
<td>See Figure 2-5 in the Virtex-II Handbook.</td>
</tr>
<tr>
<td><strong>Blue underlined text</strong></td>
<td>Hyperlink to a website (URL)</td>
<td>Go to <a href="http://www.xilinx.com">http://www.xilinx.com</a> for the latest speed files.</td>
</tr>
</tbody>
</table>
What’s New in This Release

The following design elements have been removed from the current release:

- GT10_3GIO_n
- MUXF8_D
- MUXF8_L

The following design elements have been added to the current release:

- IBUFDS_DIFF_OUT
Table of Contents

About This Guide
Guide Contents ................................................................. 3
Additional Resources ............................................................. 3
Conventions ................................................................. 4
  Typographical .......................................................... 4
  Online Document ......................................................... 4
What's New in This Release .................................................. 5

Xilinx Unified Libraries
Overview ................................................................. 17
Applicable Architectures .................................................. 18
Functional Categories ...................................................... 18
Design Elements ........................................................... 18
Schematic Examples ......................................................... 18
Naming Conventions ........................................................ 19
Attributes and Constraints .................................................. 19
Carry Logic ................................................................. 19
  Spartan-II, Spartan-IIE, Virtex, and Virtex-E ................... 19
  Virtex-II, Virtex-II Pro, Virtex-II Pro X, and Spartan-3 ...... 20
Flip-Flop, Counter, and Register Performance ...................... 20
  Unconnected Pins ..................................................... 22

Slice Count
About Configurable Logic Blocks (CLBs) ............................ 23
Slice Count for FPGA Components ..................................... 23

Architecture Specific Information
Spartan-II and Spartan-IIE ............................................... 39
Spartan-3 ........................................................................ 41
Virtex and Virtex-E .......................................................... 45
Virtex-II, Virtex-II Pro, Virtex-II Pro X ............................. 47
XC9500/XV/XL ............................................................... 51
CoolRunner XPLA3 ........................................................... 53
CoolRunner-II ................................................................. 55

Functional Categories
Arithmetic Functions ......................................................... 59
Buffers ............................................................................. 61
Comparators ................................................................. 63
Counters ................................................................. 64
Decoders ............................................................... 71
Edge Decoders ......................................................... 72
Flip-Flops ............................................................... 73
General ................................................................. 78
Input Latches ......................................................... 81
Input/Output Flip-Flops ............................................. 82
Input/Output Functions ............................................. 85
Latches ................................................................. 88
Logic Primitives ....................................................... 90
Map Elements ......................................................... 96
Memory Elements ..................................................... 97
Multiplexers .......................................................... 99
Shifters ................................................................. 100
Shift Registers ......................................................... 101

**Design Elements**

ACC1 ................................................................. 107
ACC4, 8, 16 ........................................................... 111
ADD1 ................................................................. 119
ADD4, 8, 16 ........................................................... 121
ADSU1 ................................................................. 127
ADSU4, 8, 16 ........................................................... 131
AND2-9 ................................................................. 137
AND12, 16 ............................................................... 145
BRLSHFT4, 8 ......................................................... 149
BSCAN_SPARTAN2 .................................................. 153
BSCAN_SPARTAN3 .................................................. 155
BSCAN_VIRTEX ..................................................... 157
BSCAN_VIRTEX2 .................................................... 159
BUF ................................................................. 161
BUF4, 8, 16 ............................................................ 163
BUFCF ............................................................... 165
BUFE, 4, 8, 16 ........................................................ 167
BUFG ................................................................. 171
BUFGCE .............................................................. 173
BUFGCE_1 ............................................................. 175
BUFGDLL ............................................................ 177
BUFGMUX ............................................................ 179
BUFGMUX_1 .......................................................... 181
BUFGP ............................................................... 183
BUFGSR .............................................................. 185
**BUFGTS** ................................................................. 187
**BUFT, 4, 8, 16** .................................................................. 189
**CAPTURE_SPARTAN2** .................................................... 193
**CAPTURE_SPARTAN3** ...................................................... 195
**CAPTURE_VIRTEX** .......................................................... 197
**CAPTURE_VIRTEX2** ......................................................... 199
**CB2CE, CB4CE, CB8CE, CB16CE** ..................................... 201
**CB2CLE, CB4CLE, CB8CLE, CB16CLE** .............................. 207
**CB2CLED, CB4CLED, CB8CLED, CB16CLED** ..................... 213
**CB2RE, CB4RE, CB8RE, CB16RE** ..................................... 219
**CB2RLE, CB4RLE, CB8RLE, CB16RLE** ............................... 223
**CB2X1, CB4X1, CB8X1, CB16X1** ....................................... 227
**CB2X2, CB4X2, CB8X2, CB16X2** ....................................... 233
**CBD2CE, CBD4CE, CBD8CE, CBD16CE** ............................. 239
**CBD2CLE, CBD4CLE, CBD8CLE, CBD16CLE** ...................... 243
**CBD2CLED, CBD4CLED, CBD8CLED, CBD16CLED** ............... 247
**CBD2RE, CBD4RE, CBD8RE, CBD16RE** ............................... 253
**CBD2RLE, CBD4RLE, CBD8RLE, CBD16RLE** ...................... 257
**CBD2X1, CBD4X1, CBD8X1, CBD16X1** .............................. 261
**CBD2X2, CBD4X2, CBD8X2, CBD16X2** .............................. 265
**CC8CE, CC16CE** ............................................................ 271
**CC8CLE, CC16CLE** .......................................................... 277
**CC8CLED, CC16CLED** ...................................................... 283
**CC8RE, CC16RE** .............................................................. 289
**CD4CE** ...................................................................... 295
**CD4CLE** ...................................................................... 299
**CD4RE** ...................................................................... 303
**CD4RLE** ..................................................................... 307
**CDD4CE** ..................................................................... 313
**CDD4CLE** ..................................................................... 317
**CDD4RE** ...................................................................... 321
**CDD4RLE** ..................................................................... 325
**CJ4CE, CJ5CE, CJ8CE** ..................................................... 331
**CJ4RE, CJ5RE, CJ8RE** ..................................................... 335
**CJD4CE, CJD5CE, CJD8CE** ............................................... 339
**CJD4RE, CJD5RE, CJD8RE** ............................................... 343
**CLK_DIV2,3,4,6,8,10,12,14,16** ....................................... 347
**CLK_DIV2,3,4,6,8,10,12,14,16R** ...................................... 349
**CLK_DIV2,3,4,6,8,10,12,14,16RSD** ................................. 351
**CLK_DIV2,3,4,6,8,10,12,14,16SD** ................................. 353
**CLKDLL** ...................................................................... 355
**CLKDLLLE** ................................................................... 359
<table>
<thead>
<tr>
<th>Library</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKDLLHF</td>
<td>363</td>
</tr>
<tr>
<td>COMP2, 4, 8, 16</td>
<td>367</td>
</tr>
<tr>
<td>COMPM2, 4, 8, 16</td>
<td>371</td>
</tr>
<tr>
<td>COMPMC8, 16</td>
<td>377</td>
</tr>
<tr>
<td>CR8CE, CR16CE</td>
<td>381</td>
</tr>
<tr>
<td>CRD8CE, CRD16CE</td>
<td>385</td>
</tr>
<tr>
<td>D2_4E</td>
<td>389</td>
</tr>
<tr>
<td>D3_8E</td>
<td>391</td>
</tr>
<tr>
<td>D4_16E</td>
<td>395</td>
</tr>
<tr>
<td>DCM</td>
<td>399</td>
</tr>
<tr>
<td>DEC_CC4, 8, 16</td>
<td>409</td>
</tr>
<tr>
<td>DECODE4, 8, 16</td>
<td>411</td>
</tr>
<tr>
<td>DECODE32, 64</td>
<td>415</td>
</tr>
<tr>
<td>FD</td>
<td>417</td>
</tr>
<tr>
<td>FD_1</td>
<td>421</td>
</tr>
<tr>
<td>FD4, 8, 16</td>
<td>423</td>
</tr>
<tr>
<td>FD4CE, FD8CE, FD16CE</td>
<td>425</td>
</tr>
<tr>
<td>FD4RE, FD8RE, FD16RE</td>
<td>429</td>
</tr>
<tr>
<td>FDC</td>
<td>433</td>
</tr>
<tr>
<td>FDC_1</td>
<td>437</td>
</tr>
<tr>
<td>FDCE</td>
<td>441</td>
</tr>
<tr>
<td>FDCE_1</td>
<td>445</td>
</tr>
<tr>
<td>FDCP</td>
<td>449</td>
</tr>
<tr>
<td>FDCP_1</td>
<td>453</td>
</tr>
<tr>
<td>FDCPE</td>
<td>457</td>
</tr>
<tr>
<td>FDCPE_1</td>
<td>461</td>
</tr>
<tr>
<td>FDD</td>
<td>465</td>
</tr>
<tr>
<td>FDD4,8,16</td>
<td>469</td>
</tr>
<tr>
<td>FDD4CE, FDD8CE, FDD16CE</td>
<td>471</td>
</tr>
<tr>
<td>FDD4RE, FDD8RE, FDD16RE</td>
<td>473</td>
</tr>
<tr>
<td>FDDC</td>
<td>477</td>
</tr>
<tr>
<td>FDDCE</td>
<td>481</td>
</tr>
<tr>
<td>FDDCP</td>
<td>485</td>
</tr>
<tr>
<td>FDDCPE</td>
<td>489</td>
</tr>
<tr>
<td>FDDP</td>
<td>493</td>
</tr>
<tr>
<td>FDDPE</td>
<td>497</td>
</tr>
<tr>
<td>FDDR</td>
<td>501</td>
</tr>
<tr>
<td>FDDRCPE</td>
<td>503</td>
</tr>
<tr>
<td>FDDRE</td>
<td>507</td>
</tr>
<tr>
<td>FDDRSE</td>
<td>509</td>
</tr>
<tr>
<td>FDDRS</td>
<td>513</td>
</tr>
<tr>
<td>FDDRSE</td>
<td>517</td>
</tr>
<tr>
<td>Library</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>FDDE</td>
<td>519</td>
</tr>
<tr>
<td>FDDE_1</td>
<td>521</td>
</tr>
<tr>
<td>FDDE_2</td>
<td>523</td>
</tr>
<tr>
<td>FDDE_3</td>
<td>525</td>
</tr>
<tr>
<td>FDDE_4</td>
<td>527</td>
</tr>
<tr>
<td>FDDE_5</td>
<td>531</td>
</tr>
<tr>
<td>FDDE_6</td>
<td>535</td>
</tr>
<tr>
<td>FDDE_7</td>
<td>539</td>
</tr>
<tr>
<td>FDDE_8</td>
<td>543</td>
</tr>
<tr>
<td>FDDE_9</td>
<td>547</td>
</tr>
<tr>
<td>FDDE_10</td>
<td>551</td>
</tr>
<tr>
<td>FDDE_11</td>
<td>555</td>
</tr>
<tr>
<td>FDDE_12</td>
<td>559</td>
</tr>
<tr>
<td>FDDE_13</td>
<td>563</td>
</tr>
<tr>
<td>FDDE_14</td>
<td>567</td>
</tr>
<tr>
<td>FDDE_15</td>
<td>571</td>
</tr>
<tr>
<td>FDDE_16</td>
<td>575</td>
</tr>
<tr>
<td>FDDE_17</td>
<td>579</td>
</tr>
<tr>
<td>FDDE_18</td>
<td>583</td>
</tr>
<tr>
<td>FDDE_19</td>
<td>587</td>
</tr>
<tr>
<td>FDDE_20</td>
<td>591</td>
</tr>
<tr>
<td>FDDE_21</td>
<td>595</td>
</tr>
<tr>
<td>FDDE_22</td>
<td>599</td>
</tr>
<tr>
<td>FDDE_23</td>
<td>601</td>
</tr>
<tr>
<td>FDDE_24</td>
<td>603</td>
</tr>
<tr>
<td>FDDE_25</td>
<td>607</td>
</tr>
<tr>
<td>FDDE_26</td>
<td>611</td>
</tr>
<tr>
<td>FDDE_27</td>
<td>615</td>
</tr>
<tr>
<td>FDDE_28</td>
<td>619</td>
</tr>
<tr>
<td>FDDE_29</td>
<td>623</td>
</tr>
<tr>
<td>FDDE_30</td>
<td>627</td>
</tr>
<tr>
<td>FDDE_31</td>
<td>631</td>
</tr>
<tr>
<td>FDDE_32</td>
<td>635</td>
</tr>
<tr>
<td>FDDE_33</td>
<td>637</td>
</tr>
<tr>
<td>FDDE_34</td>
<td>641</td>
</tr>
<tr>
<td>FDDE_35</td>
<td>645</td>
</tr>
<tr>
<td>FDDE_36</td>
<td>649</td>
</tr>
<tr>
<td>FDDE_37</td>
<td>653</td>
</tr>
<tr>
<td>FDDE_38</td>
<td>657</td>
</tr>
<tr>
<td>FDDE_39</td>
<td>659</td>
</tr>
<tr>
<td>FDDE_40</td>
<td>663</td>
</tr>
<tr>
<td>FDDE_41</td>
<td>665</td>
</tr>
<tr>
<td>Library Name</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>FTDCLEX</td>
<td>669</td>
</tr>
<tr>
<td>FTDCP</td>
<td>671</td>
</tr>
<tr>
<td>FTDRSE</td>
<td>673</td>
</tr>
<tr>
<td>FTDRSLE</td>
<td>677</td>
</tr>
<tr>
<td>FTP</td>
<td>681</td>
</tr>
<tr>
<td>FTEE</td>
<td>685</td>
</tr>
<tr>
<td>FTPLE</td>
<td>689</td>
</tr>
<tr>
<td>FTRSE</td>
<td>693</td>
</tr>
<tr>
<td>FTRSLE</td>
<td>697</td>
</tr>
<tr>
<td>RTSRE</td>
<td>701</td>
</tr>
<tr>
<td>RTSRLE</td>
<td>705</td>
</tr>
<tr>
<td>GND</td>
<td>709</td>
</tr>
<tr>
<td>GT_AURORA_n</td>
<td>711</td>
</tr>
<tr>
<td>GT_CUSTOM</td>
<td>735</td>
</tr>
<tr>
<td>GT_ETHERNET_n</td>
<td>745</td>
</tr>
<tr>
<td>GT_FIBRE_CHAN_n</td>
<td>769</td>
</tr>
<tr>
<td>GT_INFINIBAND_n</td>
<td>793</td>
</tr>
<tr>
<td>GT_XAUL_n</td>
<td>817</td>
</tr>
<tr>
<td>GT10_AURORA_n</td>
<td>841</td>
</tr>
<tr>
<td>GT10_AURORAX_n</td>
<td>865</td>
</tr>
<tr>
<td>GT10_CUSTOM</td>
<td>883</td>
</tr>
<tr>
<td>GT10_INFINIBAND_n</td>
<td>893</td>
</tr>
<tr>
<td>GT10_XAUL_n</td>
<td>897</td>
</tr>
<tr>
<td>GT10_10GE_n</td>
<td>923</td>
</tr>
<tr>
<td>GT10_10GFC_n</td>
<td>941</td>
</tr>
<tr>
<td>GT10_OC48_n</td>
<td>955</td>
</tr>
<tr>
<td>GT10_OC192_n</td>
<td>975</td>
</tr>
<tr>
<td>GT10_PCI_EXPRESS_n</td>
<td>989</td>
</tr>
<tr>
<td>IBUF, 4, 8, 16</td>
<td>1013</td>
</tr>
<tr>
<td>IBUF_selectIO</td>
<td>1017</td>
</tr>
<tr>
<td>IBUFDS</td>
<td>1029</td>
</tr>
<tr>
<td>IBUF, IBUF_selectIO</td>
<td>1033</td>
</tr>
<tr>
<td>IBUFGDS</td>
<td>1037</td>
</tr>
<tr>
<td>IBUF, IBUF_selectIO</td>
<td>1037</td>
</tr>
<tr>
<td>IBUFGDS_DIFF_OUT</td>
<td>1041</td>
</tr>
<tr>
<td>ICAP_VIRTEX2</td>
<td>1043</td>
</tr>
<tr>
<td>IFD, 4, 8, 16</td>
<td>1045</td>
</tr>
<tr>
<td>IFD_1</td>
<td>1049</td>
</tr>
<tr>
<td>IFDDRCPE</td>
<td>1051</td>
</tr>
<tr>
<td>IFDDRRSE</td>
<td>1055</td>
</tr>
<tr>
<td>IFDI</td>
<td>1059</td>
</tr>
<tr>
<td>IFDI_1</td>
<td>1061</td>
</tr>
<tr>
<td>IFDX, 4, 8, 16</td>
<td>1063</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>IFDX_1</td>
<td></td>
</tr>
<tr>
<td>IFDXI</td>
<td></td>
</tr>
<tr>
<td>IFDXL1</td>
<td></td>
</tr>
<tr>
<td>ILD_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>ILD_1</td>
<td></td>
</tr>
<tr>
<td>ILDI</td>
<td></td>
</tr>
<tr>
<td>ILDI_1</td>
<td></td>
</tr>
<tr>
<td>ILD_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>ILDX_1</td>
<td></td>
</tr>
<tr>
<td>ILDXI</td>
<td></td>
</tr>
<tr>
<td>ILDXL1</td>
<td></td>
</tr>
<tr>
<td>INV_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>IOBUF, IOBUF_selectIO</td>
<td></td>
</tr>
<tr>
<td>IOBUFDS</td>
<td></td>
</tr>
<tr>
<td>IOPAD_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>IPAD_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>JTAGPPC</td>
<td></td>
</tr>
<tr>
<td>KEEPER</td>
<td></td>
</tr>
<tr>
<td>LD_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>LD_1</td>
<td></td>
</tr>
<tr>
<td>LDC</td>
<td></td>
</tr>
<tr>
<td>LDC_1</td>
<td></td>
</tr>
<tr>
<td>LDCE</td>
<td></td>
</tr>
<tr>
<td>LDCE_1</td>
<td></td>
</tr>
<tr>
<td>LD4CE, LD8CE, LD16CE</td>
<td></td>
</tr>
<tr>
<td>LDCP</td>
<td></td>
</tr>
<tr>
<td>LDCP_1</td>
<td></td>
</tr>
<tr>
<td>LDCPE</td>
<td></td>
</tr>
<tr>
<td>LDCPE_1</td>
<td></td>
</tr>
<tr>
<td>LDE</td>
<td></td>
</tr>
<tr>
<td>LDE_1</td>
<td></td>
</tr>
<tr>
<td>LDG</td>
<td></td>
</tr>
<tr>
<td>LDG_4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>LDP</td>
<td></td>
</tr>
<tr>
<td>LDP_1</td>
<td></td>
</tr>
<tr>
<td>LDPE</td>
<td></td>
</tr>
<tr>
<td>LDPE_1</td>
<td></td>
</tr>
<tr>
<td>LUT1_2, 3, 4</td>
<td></td>
</tr>
<tr>
<td>LUT1_4, LUT2_4, LUT3_4, LUT4_4</td>
<td></td>
</tr>
<tr>
<td>LUT1_4, LUT2_4, LUT3_4, LUT4_4</td>
<td></td>
</tr>
<tr>
<td>M2_1</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>M2_1B1</td>
<td>1205</td>
</tr>
<tr>
<td>M2_1B2</td>
<td>1207</td>
</tr>
<tr>
<td>M2_1E</td>
<td>1209</td>
</tr>
<tr>
<td>M4_1E</td>
<td>1211</td>
</tr>
<tr>
<td>M8_1E</td>
<td>1213</td>
</tr>
<tr>
<td>M16_1E</td>
<td>1217</td>
</tr>
<tr>
<td>MULT_AND</td>
<td>1221</td>
</tr>
<tr>
<td>MULT18X18</td>
<td>1223</td>
</tr>
<tr>
<td>MULT18X18S</td>
<td>1225</td>
</tr>
<tr>
<td>MUXCY</td>
<td>1227</td>
</tr>
<tr>
<td>MUXCY_D</td>
<td>1229</td>
</tr>
<tr>
<td>MUXCY_L</td>
<td>1231</td>
</tr>
<tr>
<td>MUXF5</td>
<td>1233</td>
</tr>
<tr>
<td>MUXF5_D</td>
<td>1235</td>
</tr>
<tr>
<td>MUXF5_L</td>
<td>1237</td>
</tr>
<tr>
<td>MUXF6</td>
<td>1239</td>
</tr>
<tr>
<td>MUXF6_D</td>
<td>1241</td>
</tr>
<tr>
<td>MUXF6_L</td>
<td>1243</td>
</tr>
<tr>
<td>MUXF7</td>
<td>1245</td>
</tr>
<tr>
<td>MUXF7_D</td>
<td>1247</td>
</tr>
<tr>
<td>MUXF7_L</td>
<td>1249</td>
</tr>
<tr>
<td>MUXF8</td>
<td>1251</td>
</tr>
<tr>
<td>NAND2-9</td>
<td>1253</td>
</tr>
<tr>
<td>NAND12, 16</td>
<td>1259</td>
</tr>
<tr>
<td>NOR2-9</td>
<td>1263</td>
</tr>
<tr>
<td>NOR12, 16</td>
<td>1271</td>
</tr>
<tr>
<td>OBUF, 4, 8, 16</td>
<td>1275</td>
</tr>
<tr>
<td>OBUF_selectIO</td>
<td>1279</td>
</tr>
<tr>
<td>OBUFDS</td>
<td>1285</td>
</tr>
<tr>
<td>OBUFE, 4, 8, 16</td>
<td>1287</td>
</tr>
<tr>
<td>OBUFT, 4, 8, 16</td>
<td>1291</td>
</tr>
<tr>
<td>OBUFT_selectIO</td>
<td>1295</td>
</tr>
<tr>
<td>OBUFTDS</td>
<td>1301</td>
</tr>
<tr>
<td>OFD, 4, 8, 16</td>
<td>1305</td>
</tr>
<tr>
<td>OFD_1</td>
<td>1311</td>
</tr>
<tr>
<td>OFDDRCPE</td>
<td>1313</td>
</tr>
<tr>
<td>OFDDRRSE</td>
<td>1315</td>
</tr>
<tr>
<td>OFDDRTCPE</td>
<td>1319</td>
</tr>
<tr>
<td>OFDDRTRSE</td>
<td>1323</td>
</tr>
<tr>
<td>OFDE, 4, 8, 16</td>
<td>1327</td>
</tr>
<tr>
<td>OFDE_1</td>
<td>1331</td>
</tr>
<tr>
<td>OFDI</td>
<td>1333</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>OFDI_1</td>
<td>OFDI_1</td>
</tr>
<tr>
<td>OFDT, 4, 8, 16</td>
<td>OFDT, 4, 8, 16</td>
</tr>
<tr>
<td>OFDT_1</td>
<td>OFDT_1</td>
</tr>
<tr>
<td>OFDX, 4, 8, 16</td>
<td>OFDX, 4, 8, 16</td>
</tr>
<tr>
<td>OFDX</td>
<td>OFDX</td>
</tr>
<tr>
<td>OFDXI</td>
<td>OFDXI</td>
</tr>
<tr>
<td>OFDXI_1</td>
<td>OFDXI_1</td>
</tr>
<tr>
<td>OPAD, 4, 8, 16</td>
<td>OPAD, 4, 8, 16</td>
</tr>
<tr>
<td>OR2-9</td>
<td>OR2-9</td>
</tr>
<tr>
<td>OR12, 16</td>
<td>OR12, 16</td>
</tr>
<tr>
<td>ORCY</td>
<td>ORCY</td>
</tr>
<tr>
<td>PPC405</td>
<td>PPC405</td>
</tr>
<tr>
<td>PULLDOWN</td>
<td>PULLDOWN</td>
</tr>
<tr>
<td>PULLUP</td>
<td>PULLUP</td>
</tr>
<tr>
<td>RAM16X1D</td>
<td>RAM16X1D</td>
</tr>
<tr>
<td>RAM16X1D_1</td>
<td>RAM16X1D_1</td>
</tr>
<tr>
<td>RAM16X1S</td>
<td>RAM16X1S</td>
</tr>
<tr>
<td>RAM16X1S_1</td>
<td>RAM16X1S_1</td>
</tr>
<tr>
<td>RAM16X2D</td>
<td>RAM16X2D</td>
</tr>
<tr>
<td>RAM16X2S</td>
<td>RAM16X2S</td>
</tr>
<tr>
<td>RAM16X4D</td>
<td>RAM16X4D</td>
</tr>
<tr>
<td>RAM16X4S</td>
<td>RAM16X4S</td>
</tr>
<tr>
<td>RAM16X8D</td>
<td>RAM16X8D</td>
</tr>
<tr>
<td>RAM16X8S</td>
<td>RAM16X8S</td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>RAM32X1D</td>
</tr>
<tr>
<td>RAM32X1D_1</td>
<td>RAM32X1D_1</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>RAM32X1S</td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>RAM32X1S_1</td>
</tr>
<tr>
<td>RAM32X2S</td>
<td>RAM32X2S</td>
</tr>
<tr>
<td>RAM32X4S</td>
<td>RAM32X4S</td>
</tr>
<tr>
<td>RAM32X8S</td>
<td>RAM32X8S</td>
</tr>
<tr>
<td>RAM64X1D</td>
<td>RAM64X1D</td>
</tr>
<tr>
<td>RAM64X1D_1</td>
<td>RAM64X1D_1</td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>RAM64X1S</td>
</tr>
<tr>
<td>RAM64X1S_1</td>
<td>RAM64X1S_1</td>
</tr>
<tr>
<td>RAM64X2S</td>
<td>RAM64X2S</td>
</tr>
<tr>
<td>RAM128X1S</td>
<td>RAM128X1S</td>
</tr>
<tr>
<td>RAM128X1S_1</td>
<td>RAM128X1S_1</td>
</tr>
<tr>
<td>RAMB4_Sn</td>
<td>RAMB4_Sn</td>
</tr>
<tr>
<td>RAMB4_Sm_Sn</td>
<td>RAMB4_Sm_Sn</td>
</tr>
<tr>
<td>RAMB16_Sn</td>
<td>RAMB16_Sn</td>
</tr>
<tr>
<td>RAMB16_Sm_Sn</td>
<td>RAMB16_Sm_Sn</td>
</tr>
<tr>
<td>Component</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>ROC</td>
<td>1557</td>
</tr>
<tr>
<td>ROCBUF</td>
<td>1559</td>
</tr>
<tr>
<td>ROM16X1</td>
<td>1561</td>
</tr>
<tr>
<td>ROM32X1</td>
<td>1563</td>
</tr>
<tr>
<td>ROM64X1</td>
<td>1565</td>
</tr>
<tr>
<td>ROM128X1</td>
<td>1567</td>
</tr>
<tr>
<td>ROM256X1</td>
<td>1569</td>
</tr>
<tr>
<td>SOP3-4</td>
<td>1571</td>
</tr>
<tr>
<td>SR4CE, SR8CE, SR16CE</td>
<td>1573</td>
</tr>
<tr>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>1577</td>
</tr>
<tr>
<td>SR4CLED, SR8CLED, SR16CLE</td>
<td>1581</td>
</tr>
<tr>
<td>SR4RE, SR8RE, SR16RE</td>
<td>1587</td>
</tr>
<tr>
<td>SR4RLE, SR8RLE, SR16RLE</td>
<td>1591</td>
</tr>
<tr>
<td>SR4RLED, SR8RLED, SR16RLED</td>
<td>1595</td>
</tr>
<tr>
<td>SRD4CE, SRD8CE, SRD16CE</td>
<td>1601</td>
</tr>
<tr>
<td>SRD4CLE, SRD8CLE, SRD16CLE</td>
<td>1605</td>
</tr>
<tr>
<td>SRD4CLED, SRD8CLED, SRD16CLED</td>
<td>1609</td>
</tr>
<tr>
<td>SRD4RE, SRD8RE, SRD16RE</td>
<td>1613</td>
</tr>
<tr>
<td>SRD4RLE, SRD8RLE, SRD16RLE</td>
<td>1617</td>
</tr>
<tr>
<td>SRD4RLED, SRD8RLED, SRD16RLED</td>
<td>1621</td>
</tr>
<tr>
<td>SRL16</td>
<td>1625</td>
</tr>
<tr>
<td>SRL16_1</td>
<td>1629</td>
</tr>
<tr>
<td>SRL16E</td>
<td>1633</td>
</tr>
<tr>
<td>SRL16E_1</td>
<td>1637</td>
</tr>
<tr>
<td>SRLC16</td>
<td>1641</td>
</tr>
<tr>
<td>SRLC16_1</td>
<td>1645</td>
</tr>
<tr>
<td>SRLC16E</td>
<td>1647</td>
</tr>
<tr>
<td>SRLC16E_1</td>
<td>1651</td>
</tr>
<tr>
<td>STARTBUF_architecture</td>
<td>1655</td>
</tr>
<tr>
<td>STARTUP_SPARTAN2</td>
<td>1657</td>
</tr>
<tr>
<td>STARTUP_SPARTAN3</td>
<td>1659</td>
</tr>
<tr>
<td>STARTUP_VIRTEX</td>
<td>1663</td>
</tr>
<tr>
<td>STARTUP_VIRTEX2</td>
<td>1667</td>
</tr>
<tr>
<td>TOC</td>
<td>1671</td>
</tr>
<tr>
<td>TOCBUF</td>
<td>1673</td>
</tr>
<tr>
<td>VCC</td>
<td>1675</td>
</tr>
<tr>
<td>UPAD</td>
<td>1677</td>
</tr>
<tr>
<td>XNOR2-9</td>
<td>1679</td>
</tr>
<tr>
<td>XOR2-9</td>
<td>1685</td>
</tr>
<tr>
<td>XORCY</td>
<td>1691</td>
</tr>
<tr>
<td>XORCY_D</td>
<td>1693</td>
</tr>
<tr>
<td>XORCY_L</td>
<td>1695</td>
</tr>
</tbody>
</table>
Xilinx Unified Libraries

This chapter describes the Unified Libraries and the applicable device architectures for each library. It also briefly discusses the contents of the other chapters, the general naming conventions, and performance issues.

This chapter consists of the following major sections.

- “Overview”
- “Applicable Architectures”
- “Functional Categories”
- “Design Elements”
- “Schematic Examples”
- “Naming Conventions”
- “Attributes and Constraints”
- “Carry Logic”
- “Flip-Flop, Counter, and Register Performance”
- “Unconnected Pins”

Overview

Xilinx maintains software libraries with thousands of functional design elements (primitives and macros) for different device architectures. New functional elements are assembled with each release of development system software. The catalog of design elements is known as the Unified Libraries. Elements in these libraries are common to all Xilinx device architectures. This “unified” approach means that you can use your circuit design created with “unified” library elements across all current Xilinx device architectures that recognize the element you are using.

Elements that exist in multiple architectures look and function the same, but their implementations might differ to make them more efficient for a particular architecture. A separate library still exists for each architecture (or architectural group) and common symbols are duplicated in each one, which is necessary for simulation (especially board level) where timing depends on a particular architecture.

If you have active designs that were created with former Xilinx library primitives or macros, you may need to change references to the design elements that you were using to reflect the Unified Libraries elements.

The Libraries Guide describes the primitive and macro logic elements available in the Unified Libraries for the Xilinx FPGA and CPLD devices. Common logic functions can be implemented with these elements and more complex functions can be built by combining macros and primitives. Several hundred design elements (primitives and
Xilinx Unified Libraries

macros) are available across multiple device architectures, providing a common base for programmable logic designs.

This libraries guide provides a functional selection guide and describes the design elements.

Applicable Architectures

Design elements for the Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex -II, Virtex-II Pro, Virtex-II Pro X, XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II libraries are included in the Xilinx Unified Libraries. Each library supports specific device architectures. For detailed information on the architectural families referenced below and the devices in each, see the current version of The Programmable Logic Data Book (an online version is available from the Xilinx web site, http://support.xilinx.com).

Functional Categories

The functional categories list the available elements in each category along with a brief description of each element and an applicability table identifying which libraries (Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex -II, Virtex-II Pro, Virtex-II Pro X, XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II) contain the element.

Design Elements

Design elements are organized in alphanumeric order, with all numeric suffixes in ascending order. For example, FDR precedes FDRS, and ADD4 precedes ADD8, which precedes ADD16.

The following information is provided for each library element, where applicable:

- Graphic symbol
- Applicability table (with primitive versus macro identification)
- Functional description
- Truth table
- Schematic for macros
- VHDL and Verilog instantiation and inference code
- Commonly used constraints

Schematic Examples

Schematics are included for each library if the implementation differs.

Design elements with bussed or multiple I/O pins (2-, 4-, 8-, 16-bit versions) typically include just one schematic -- generally the 8-bit version. When only one schematic is included, implementation of the smaller and larger elements differs only in the number of sections. In cases where an 8-bit version is very large, an appropriate smaller element serves as the schematic example.
Naming Conventions

Examples of the general naming conventions for the unified library elements are shown in the following figures.

Attributes and Constraints

Attributes and constraints are instructions placed on components or nets to indicate their placement, implementation, naming, directionality, and so forth. The Constraints Guide provides information on all attributes and constraints.

Carry Logic

The Spartan-II, Spartan-IIIE, Virtex, and Virtex-II architectures include dedicated carry logic components.

Spartan-II, Spartan-IIIE, Virtex, and Virtex-E

Carry Logic for Spartan-II, Spartan-IIIE, Virtex, and Virtex-E is a simple structure associated with each look-up table. The design entry library contains the following dedicated carry logic primitives: MULT_AND, MUXCY, MUXCY_D, MUXCY_L, XORCY, XORCY_D, and XORCY_L. The function performed is determined by their
connectivity and the contents of the look-up table. For an example of how to use carry logic, see “CC8CE, CC16CE”.


Virtex-II, Virtex-II Pro, Virtex-II Pro X, and Spartan-3

The dedicated carry logic primitives for Virtex-II, Virtex-II Pro, Virtex-II Pro X, and Spartan-3 are MULT_AND, MUXCY, MUXCY_D, MUXCY_L, XORCY, XORCY_D, and XORCY_L.

ORCY can only be used exclusively with Virtex-II, Virtex-II Pro, and Virtex-II Pro X.


Flip-Flop, Counter, and Register Performance

All counter, register, and storage functions derived from the flip-flops are available in the Configurable Logic Blocks (CLBs).

The D flip-flop is the basic building block for all architectures. Differences occur from the availability of asynchronous Clear (CLR) and Preset (PRE) inputs, and the source of the synchronous control signals, such as Clock Enable (CE), Clock (C), Load enable (L), synchronous Reset (R), and synchronous Set (S). The basic flip-flop configuration for each architecture follows.

The basic XC9000 flip-flops have both Clear and Preset inputs.

Virtex and Spartan-II have two basic flip-flop types. One has both Clear and Preset inputs and one has both asynchronous and synchronous control functions.
The asynchronous and synchronous control functions, when used, have a priority that is consistent across all devices and architectures. These inputs can be either active-High or active-Low as defined by the macro. The priority, from highest to lowest, is as follows.

- Asynchronous Clear (CLR)
- Asynchronous Preset (PRE)
- Synchronous Set (S)
- Synchronous Reset (R)
- Clock Enable (CE)

**Note:** The asynchronous CLR and PRE inputs, by definition, have priority over all the synchronous control and clock inputs.

For FPGA families, the Clock Enable (CE) function is implemented using two different methods in the Xilinx Unified Libraries; both are shown in the following figure.

- In method 1, CE is implemented by connecting the CE pin of the macro directly to the dedicated Enable Clock (EC) pin of the internal Configurable Logic Block (CLB) flip-flop. This allows one CE per CLB. CE takes precedence over the L, S, and R inputs. All flip-flops with asynchronous clear or preset use this method.
- In method 2, CE is implemented using function generator logic. This allows two CE per CLB. CE has the same priority as the L, S, and R inputs. All flip-flops with synchronous set or reset use this method.

The method used in a particular macro is indicated by the inclusion of asynchronous clear, asynchronous preset, synchronous set, or synchronous reset in the macro's description.
Clock Enable Implementation Methods

Unconnected Pins

Xilinx recommends that you *always* connect input pins in your designs. This ensures that front end simulation functionally matches back end timing simulation. If an input pin is left unconnected, mapper errors may result.

If an output pin is left unconnected in your design, the corresponding function is trimmed. If the component has only one output, the entire component is trimmed. If the component has multiple outputs, the portion that drives the output is trimmed. As an example of the latter case, if the overflow pin (OFL) in an adder macro is unconnected, the logic that generates that term is trimmed, but the rest of the adder is retained (assuming all of the sum outputs are connected).
Slice Count

This chapter contains the following sections.

- About Configurable Logic Blocks (CLBs)
- Slice Count for FPGA Components

About Configurable Logic Blocks (CLBs)

Configurable Logic Blocks (CLBs) implement most of the logic in an FPGA.

Each Virtex, Virtex-E and Spartan-II, Spartan-IIE CLB contains two slices. Each Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X CLB contains four slices. In the following table, the numbers for Spartan-II, Spartan-IIE, Virtex, Virtex-E, Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X are the number of slices required to implement the component.

The Slice Count table lists FPGA design elements in alphanumeric order with the number of CLBs or slices needed for their implementation in each applicable library.

**Note:** This information is for reference only. The actual count could vary, depending upon the switch settings of the implementation tools; for example, the effort level in PAR (Place and Route) or usage of the components with other components.

The asterisk for the RAM16X1D and RAM16X1D_1 in the Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X columns indicates that these design elements consume 1/2 of two slices.

The double asterisks for design elements indicate that these primitives cannot be used by themselves. However, there is only one available per slice.

Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>ACC8</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>ACC16</td>
<td>17</td>
<td>18</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>ADD4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>ADD8</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ADD16</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>ADSU4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>ADSU8</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ADSU16</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AND9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AND12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AND16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>BRLSHIFT4</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>BRLSHIFT8</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>BSCAN_SPARTAN2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BSCAN_VIRTEX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BSCAN_VIRTEX2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUF4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUF8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFX6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUF4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFX8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFG</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFGE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFGE_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFGDLL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFGMUX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFGMUX_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFGP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFT4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFT8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFT16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAPTURE_SPARTAN2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAPTURE_VIRTEX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE_VIRTEX2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CB2CE</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>CB2CLE</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>CB2CLED</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>CB2RE</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>CB4CE</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>CB4CLE</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CB4CLED</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>CB4RE</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>CB8CE</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>CB8CLE</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>CB8CLED</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>CB8RE</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>CB16CE</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>CB16CLE</td>
<td>18</td>
<td>19</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>CB16CLED</td>
<td>24</td>
<td>25</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>CB16RE</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>CC8CE</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>CC8CLE</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>CC8CLED</td>
<td>9</td>
<td>17</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>CC8RE</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>CC16CE</td>
<td>16</td>
<td>9</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>CC16CLE</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>CC16CLED</td>
<td>17</td>
<td>33</td>
<td>17</td>
<td>33</td>
</tr>
<tr>
<td>CC16RE</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>CD4CE</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>CD4CLE</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CD4RE</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>CD4RLE</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>CJ4CE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>CJ4RE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>CJ5CE</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>CJ5RE</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>CJ8CE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CJ8RE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CLKDLL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CLKDLLLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
# Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKDLLHF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COMP2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>COMP4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>COMP8</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>COMP16</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>COMPM2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>COMPM4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>COMPM8</td>
<td>11</td>
<td>13</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>COMPM16</td>
<td>24</td>
<td>32</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>COMPMC8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>COMPMC16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>CR8CE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>CR16CE</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>D2_4E</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D3_8E</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>D4_16E</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>DCM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DEC_CC4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC_CC8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC_CC16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DECODE4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DECODE8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DECODE16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DECODE32</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>DECODE64</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FD</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FD_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FD4CE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>FD4RE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>FD8CE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>FD8RE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>FD16CE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FD16RE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FDC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDC_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDCE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDCE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDCP_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDCPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDCPE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDDRCPE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FDDRRSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FDE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDP_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDPE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDR_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDRE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDRE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDRS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDRS_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDRSE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDRSE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDS_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDSE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FDSE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FJKC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FJKCE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FJKP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FJKPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FJKRSE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FJKSRE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FMAP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FTC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTCE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTCLE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTCLEX</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPLE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTRSE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTRSLE</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>FTSRE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FTSRSL</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>GND</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT_AURORA_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT_CUSTOM_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT_ETHERNET_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT_FIBRE_CHAN_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT_INFINIBAND_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT_HAUI_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_AURORA_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_AURORAX_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_CUSTOM_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_INFINIBAND_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_HAUI_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_10GE_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_10GFC_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_OC48_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GT10_OC192_n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUF4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUF8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUF16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUF_selectIO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUFDS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUG</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUFG_selectIO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUFGDS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IBUFGDS_DIFF_OUT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ICAP_VIRTEX2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFD_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFDDRCPE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDDRRSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDI_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDX4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDX8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDX16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDX_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDXI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IFDXI_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ILD_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ILD4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ILD8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>ILD16</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>ILD1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD1_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD1_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ILD1_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INV</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INV4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INV8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INV16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>IOBUF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IOBUF_selectIO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IOPAD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IOPAD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IOPAD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IOPAD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IPAD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IPAD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPAD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IPAD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JTAGPPC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>KEEPER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LD</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LD_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LD4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>LD8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>LD16</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>LD4CE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>LD8CE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>LD16CE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>LDC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDC_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDCE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDCE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDCP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDCP_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDCPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDCPE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDP_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LDPE_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT1_D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT2_D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT3_D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT4_D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT1_L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT2_L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LUT3_L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT4_L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M2_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M2_1B1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M2_1B2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M2_1E</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M4_1E</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M8_1E</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>M16_1E</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>MULT_AND **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MULT18X18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MULT18X18S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXCY **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXCY_D **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXCY_L **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF5 **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF5_D **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF5_L **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF6 **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF6_D **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF6_L **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF7 **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF7_D **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF7_L **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF8 **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MUXF8_D **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NAND2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NAND9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NAND12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NAND16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NOR2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
# Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOR3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOR4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOR5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOR6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOR7</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NOR8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NOR9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NOR12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NOR16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>OBUF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUF4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUF8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUF16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUF_selectIO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFD5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFE4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFE8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFE16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFT4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFT8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFT16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFT_selectIO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OBUFTD5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFD_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDDRCPE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDDRRSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDDRTPCE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDDRTRSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDE_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDE4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFDE8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDE16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDI1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDT_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDT4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDT8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDT16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDX4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDX8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDX16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDX_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OFDXX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OPAD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OPAD4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OPAD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OPAD16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OR2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OR3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OR4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OR5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OR6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OR7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OR8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>OR9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>OR12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>OR16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ORCY **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PPC405</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PULLDOWN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PULLUP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAM16X1D</td>
<td>1</td>
<td>2*</td>
<td>1</td>
<td>2*</td>
</tr>
<tr>
<td>RAM16X1D_1</td>
<td>1</td>
<td>2*</td>
<td>1</td>
<td>2*</td>
</tr>
<tr>
<td>RAM16X1S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM16X1S_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RAM16X2D</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>RAM16X2S</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>RAM16X4D</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>RAM16X4S</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>RAM16X8D</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>RAM16X8S</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>RAM32X1D_1</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RAM32X2S</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>RAM32X4S</td>
<td>8</td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>RAM32X8S</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>RAM64X1D</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>RAM64X1D_1</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>RAM64X1S_1</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>RAM64X2S</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>RAM128X1S</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>RAM128X1S_1</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>RAMB4_Sn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAMB4_Sm_Sn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAMB16_Sn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAMB16_Sm_Sn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ROM16X1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ROM32X1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ROM64X1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ROM128X1</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>ROM256X1</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>SOP3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SOP4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SR4CE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>SR4CLE</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SR4CLED</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>SR4RE</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>SR4RLE</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR4RLED</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>SR8CE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>SR8CLE</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>SR8CLED</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>SR8RE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>SR8RLE</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>SR8RLED</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>SR16CE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>SR16CLE</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>SR16CLED</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>SR16RE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>SR16RLE</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>SR16RLED</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>SRL16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SRL16_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SRL16E</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SRL16E_1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SRLC16</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>SRLC16_1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>SRLC16E</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>SRLC16E_1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>STARTUP_SPARtan2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STARTUP_VIRtex</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STARTUP_VIRtex2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UPAD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VCC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XNOR2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XNOR3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XNOR4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XNOR5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XNOR6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XNOR7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XNOR8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>XNOR9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>XOR2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Slice Count for FPGA Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>XOR9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>XORCY **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XORCY_D **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XORCY_L **</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

* The RAM16X1D and RAM16X1D_1 consume 1/2 of two slices.

** These primitives cannot be used by themselves. However, there is only one available per slice.
Architecture Specific Information

The following sections list the design elements that can be used with supported architectures.

- Spartan-II and Spartan-IIE
- Spartan-3
- Virtex and Virtex-E
- Virtex-II, Virtex-II Pro, Virtex-II Pro X
- XC9500/XV/XL
- CoolRunner XPLA3
- CoolRunner-II

To access lists of the constraints associated with each of these architectures, see "Architecture Specific Constraints," in the Xilinx Constraints Guide.
The following table indicates the supported design elements for Spartan-II and Spartan-IIE. For a complete description of these architectures, see the Product Data Sheets (http://www.xilinx.com/partinfo/databook.htm#spartan).

**Spartan-II, Spartan-IIE Design Elements**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC4, 8, 16</td>
<td>ADD4, 8, 16</td>
<td>ADSU4, 8, 16</td>
</tr>
<tr>
<td>AND2-9</td>
<td>AND12, 16</td>
<td>BRLSHTF4, 8</td>
</tr>
<tr>
<td>BSCAN_SPARTAN2</td>
<td>BUF</td>
<td>BUFCF</td>
</tr>
<tr>
<td>BUFE, 4, 8, 16</td>
<td>BUFG</td>
<td>BUFGDLL</td>
</tr>
<tr>
<td>BUFGP</td>
<td>BUFT, 4, 8, 16</td>
<td>CAPTURE_SPARTAN2</td>
</tr>
<tr>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
<td>CB2CLED, CB4CLED, CB8CLED, CB16CLE</td>
</tr>
<tr>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
<td>CC8CE, CC16CE</td>
<td>CC8CLE, CC16CLE</td>
</tr>
<tr>
<td>CC8CLED, CC16CLE</td>
<td>CC8RE, CC16RE</td>
<td>CD4CLE</td>
</tr>
<tr>
<td>CD4CLE</td>
<td>CD4RE</td>
<td>CD4RLE</td>
</tr>
<tr>
<td>CJ4CE, CJ5CE, CJ8CE</td>
<td>CJ4RE, CJ5RE, CJ8RE</td>
<td>CLKDLL, CLKDLLE (Spartan-IIE only)</td>
</tr>
<tr>
<td>CLKDLLHF</td>
<td>COMP2, 4, 8, 16</td>
<td>COMPM2, 4, 8, 16</td>
</tr>
<tr>
<td>COMPMMC8, 16</td>
<td>CR8CE, CR16CE</td>
<td>D2_4E</td>
</tr>
<tr>
<td>D3_8E</td>
<td>D4_16E</td>
<td>DEC_CC4, 8, 16</td>
</tr>
<tr>
<td>DECODE4, 8, 16</td>
<td>DECODE32, 64</td>
<td>FD</td>
</tr>
<tr>
<td>FD_1</td>
<td>FD4CE, FD8CE, FD16CE</td>
<td>FD4RE, FD8RE, FD16RE</td>
</tr>
<tr>
<td>FD</td>
<td>FDC</td>
<td>FDCE</td>
</tr>
<tr>
<td>FDC_1</td>
<td>FDCP</td>
<td>FDCE_1</td>
</tr>
<tr>
<td>FDCPE</td>
<td>FDCPE_1</td>
<td>FDE</td>
</tr>
<tr>
<td>FDE_1</td>
<td>FDP</td>
<td>FDP_1</td>
</tr>
<tr>
<td>FDPE</td>
<td>FDPE_1</td>
<td>FDR</td>
</tr>
<tr>
<td>FDR_1</td>
<td>FDRE</td>
<td>FDR_1</td>
</tr>
<tr>
<td>FDRS</td>
<td>FDRS_1</td>
<td>FDRSE</td>
</tr>
<tr>
<td>FDRSE_1</td>
<td>FDS</td>
<td>FJS</td>
</tr>
<tr>
<td>FDSE</td>
<td>FDSE_1</td>
<td>FJKC</td>
</tr>
<tr>
<td>FJKCE</td>
<td>FJKP</td>
<td>FJKPE</td>
</tr>
<tr>
<td>FJKRSE</td>
<td>FJKSRE</td>
<td>FMAP</td>
</tr>
<tr>
<td>FTC</td>
<td>FTCE</td>
<td>FTCE</td>
</tr>
<tr>
<td>FTCLEX</td>
<td>FTP</td>
<td>FTP</td>
</tr>
<tr>
<td>FTPL</td>
<td>FTRSE</td>
<td>FTRSE</td>
</tr>
<tr>
<td>FTSRE</td>
<td>FTSRLE</td>
<td>GND</td>
</tr>
<tr>
<td>IBUF, 4, 8, 16</td>
<td>IBUF_selectIO</td>
<td>IBUFG, IBUFG_selectIO</td>
</tr>
<tr>
<td>IFD, 4, 8, 16</td>
<td>IFD_1</td>
<td>IFDI</td>
</tr>
<tr>
<td>IFDL_1</td>
<td>IFDX, 4, 8, 16</td>
<td>IFDX_1</td>
</tr>
<tr>
<td>IFDXI</td>
<td>IFDXI_1</td>
<td>ILD, 4, 8, 16</td>
</tr>
<tr>
<td>ILD_1</td>
<td>ILDI</td>
<td>ILD_1</td>
</tr>
<tr>
<td>ILDX, 4, 8, 16</td>
<td>ILDX_1</td>
<td>ILDXI</td>
</tr>
</tbody>
</table>
## Spartan-II, Spartan-IIE Design Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILDXI_1</td>
<td>IN, 4, 8, 16</td>
<td>IOBUF, IOBUF_selectIO</td>
</tr>
<tr>
<td>IOPAD, 4, 8, 16</td>
<td>IPAD, 4, 8, 16</td>
<td>KEEPER</td>
</tr>
<tr>
<td>LD</td>
<td>LD_1</td>
<td>LD4, 8, 16</td>
</tr>
<tr>
<td>LDC</td>
<td>LDC_1</td>
<td>LDCE</td>
</tr>
<tr>
<td>LDCE_1</td>
<td>LD4CE, LD8CE, LD16CE</td>
<td>LDCP</td>
</tr>
<tr>
<td>LDCP_1</td>
<td>LDCPE</td>
<td>LDCP_1</td>
</tr>
<tr>
<td>LDE</td>
<td>LDE_1</td>
<td>LDP</td>
</tr>
<tr>
<td>LDP_1</td>
<td>LDP</td>
<td>LDU</td>
</tr>
<tr>
<td>LUT1, 2, 3, 4</td>
<td>LUT1_D, LUT2_D, LUT3_D, LUT4_D</td>
<td>LUT1_L, LUT2_L, LUT3_L, LUT4_L</td>
</tr>
<tr>
<td>M2_1</td>
<td>M2_1BI</td>
<td>M2_1B2</td>
</tr>
<tr>
<td>M2_1E</td>
<td>M4_1E</td>
<td>M8_1E</td>
</tr>
<tr>
<td>M16_1E</td>
<td>MULT_AND</td>
<td>MUXCY</td>
</tr>
<tr>
<td>MUXCY_D</td>
<td>MUXCY_L</td>
<td>MUXF5</td>
</tr>
<tr>
<td>MUXF5_D</td>
<td>MUXF5_L</td>
<td>MUXF6</td>
</tr>
<tr>
<td>MUXF6_D</td>
<td>MUXF6_L</td>
<td>NAND2-9</td>
</tr>
<tr>
<td>NAND12, 16</td>
<td>NOR2-9</td>
<td>NOR12, 16</td>
</tr>
<tr>
<td>OBUF, 4, 8, 16</td>
<td>OBUF_selectIO</td>
<td>OBUFE, 4, 8, 16</td>
</tr>
<tr>
<td>OBUFT, 4, 8, 16</td>
<td>OBUFT_selectIO</td>
<td>OFD, 4, 8, 16</td>
</tr>
<tr>
<td>OFD1</td>
<td>OFDE, 4, 8, 16</td>
<td>OFDE_1</td>
</tr>
<tr>
<td>OFDI</td>
<td>OFDI_1</td>
<td>OFDT, 4, 8, 16</td>
</tr>
<tr>
<td>OFDT_1</td>
<td>OFDX, 4, 8, 16</td>
<td>OFDX_1</td>
</tr>
<tr>
<td>OFDXI</td>
<td>OFDXI_1</td>
<td>OPAD, 4, 8, 16</td>
</tr>
<tr>
<td>OR2-9</td>
<td>OR12, 16</td>
<td>PULLDOWN</td>
</tr>
<tr>
<td>PULLUP</td>
<td>RAM16X1D</td>
<td>RAM16X1D_1</td>
</tr>
<tr>
<td>RAM16X1S</td>
<td>RAM16X1S_1</td>
<td>RAM16X2D</td>
</tr>
<tr>
<td>RAM16X2S</td>
<td>RAM16X4D</td>
<td>RAM16X4S</td>
</tr>
<tr>
<td>RAM16X8D</td>
<td>RAM16X8S</td>
<td>RAM32X1S</td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>RAM32X2S</td>
<td>RAM32X4S</td>
</tr>
<tr>
<td>RAM32X8S</td>
<td>RAMB4_Sn</td>
<td>RAMB4_Sm_Sn</td>
</tr>
<tr>
<td>ROC</td>
<td>ROCBUF</td>
<td>ROM16X1</td>
</tr>
<tr>
<td>ROM32X1</td>
<td>SOP3-4</td>
<td>SR4CE, SR8CE, SR16CE</td>
</tr>
<tr>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>SR4CLED, SR8CLED, SR16CLED</td>
<td>SR4RE, SR8RE, SR16RE</td>
</tr>
<tr>
<td>SR4RLE, SR8RLE, SR16RLE</td>
<td>SR4RLED, SR8RLED, SR16RLED</td>
<td>SRL16</td>
</tr>
<tr>
<td>SRL16, 1</td>
<td>SRL16E</td>
<td>SRL16E_1</td>
</tr>
<tr>
<td>STARTBUF_architecture</td>
<td>STARTUP_SPARTAN2</td>
<td>TOC</td>
</tr>
<tr>
<td>TOCBUF</td>
<td>UPAD</td>
<td>VCC</td>
</tr>
<tr>
<td>XNOR2-9</td>
<td>XOR2-9</td>
<td>XORCY</td>
</tr>
<tr>
<td>XORCY_D</td>
<td>XORCY_L</td>
<td></td>
</tr>
</tbody>
</table>
The table below indicates the supported design elements for Spartan-3.

**Spartan-3 Design Elements**

<table>
<thead>
<tr>
<th>Spartan-3 Design Elements</th>
<th>Spartan-3 Design Elements</th>
<th>Spartan-3 Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC4, 8, 16</td>
<td>ADD4, 8, 16</td>
<td>ADSU4, 8, 16</td>
</tr>
<tr>
<td>AND2-9</td>
<td>AND12, 16</td>
<td>BRLSHFT4, 8</td>
</tr>
<tr>
<td>BSCAN_SPARTAN3</td>
<td>BUF</td>
<td>BUFGFC</td>
</tr>
<tr>
<td>BUFG</td>
<td>BUFGCE</td>
<td>BUFGCE_1</td>
</tr>
<tr>
<td>BUFGDLL</td>
<td>BUFGMUX</td>
<td>BUFGMUX_1</td>
</tr>
<tr>
<td>BUFGP</td>
<td>CAPTURE_SPARTAN3</td>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
</tr>
<tr>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
<td>CB2CLED, CB4CLED, CB8CLED, CB16CLE</td>
</tr>
<tr>
<td>CC8CE, CC16CE</td>
<td>CC8CLE, CC16CLE</td>
<td>CC8CLED, CC16CLED</td>
</tr>
<tr>
<td>CC8RE, CC16RE</td>
<td>CD4CE</td>
<td>CD4CLE</td>
</tr>
<tr>
<td>CD4RE</td>
<td>CD4RLE</td>
<td>CJ4CE, CJ5CE, CJ8CE</td>
</tr>
<tr>
<td>CJ4RE, CJ5RE, CJ8RE</td>
<td>COMP2, 4, 8, 16</td>
<td>COMP2M2, 4, 8, 16</td>
</tr>
<tr>
<td>COMP2MCS, 16</td>
<td>CR8CE, CR16CE</td>
<td>D2_4E</td>
</tr>
<tr>
<td>D3_8E</td>
<td>D4_16E</td>
<td>DCM</td>
</tr>
<tr>
<td>DEC_CC4, 8, 16</td>
<td>DECODE4, 8, 16</td>
<td>DECODE32, 64</td>
</tr>
<tr>
<td>FD</td>
<td>FD_.1_</td>
<td>FD4CE, FD8CE, FD16CE</td>
</tr>
<tr>
<td>FD4RE, FD8RE, FD16RE</td>
<td>FDC</td>
<td>FDC_.1</td>
</tr>
<tr>
<td>FDCE</td>
<td>FDCE_.1_</td>
<td>FDCEP</td>
</tr>
<tr>
<td>FDCP_.1_</td>
<td>FDCPE</td>
<td>FDCPE_.1_</td>
</tr>
<tr>
<td>FDDRCPE</td>
<td>FDDRSE</td>
<td>FDE</td>
</tr>
<tr>
<td>FDE_.1_</td>
<td>FDP</td>
<td>FDP_.1_</td>
</tr>
<tr>
<td>FDPPE_.1_</td>
<td>FDR</td>
<td>FDR_.1_</td>
</tr>
<tr>
<td>FDRE_1</td>
<td>FDRE_.1_</td>
<td>FDRS</td>
</tr>
<tr>
<td>FDRS_1_</td>
<td>FDRSE</td>
<td>FDRSE_.1_</td>
</tr>
<tr>
<td>FDS_1_</td>
<td>FDS_.1_</td>
<td>FDS</td>
</tr>
<tr>
<td>FDSE_1</td>
<td>FJKC</td>
<td>FJKCE</td>
</tr>
<tr>
<td>FJKP_1</td>
<td>FJKPE</td>
<td>FJKRSE</td>
</tr>
<tr>
<td>FJKSRE_1</td>
<td>FMAP</td>
<td>FTC</td>
</tr>
<tr>
<td>FTC_E_1</td>
<td>FTCLE</td>
<td>FTCLEX</td>
</tr>
<tr>
<td>FTP_1_</td>
<td>FTP</td>
<td>FTPLE</td>
</tr>
<tr>
<td>FTRSE_1</td>
<td>FTRSL_E</td>
<td>FTSRE</td>
</tr>
<tr>
<td>FTSRLE_1</td>
<td>GND</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>IBUFDS_1</td>
<td>IBUF, IBUFG_selectIO</td>
<td>IBUFGDS</td>
</tr>
<tr>
<td>IFD, 4, 8, 16</td>
<td>IFD_.1_</td>
<td>IFDRCPE</td>
</tr>
<tr>
<td>IFDDRSE_1</td>
<td>IFDI_1</td>
<td>IFDI_1</td>
</tr>
<tr>
<td>IFDIX_1</td>
<td>IFDIX_1</td>
<td>IFDIX_1</td>
</tr>
<tr>
<td>IFDIXI_1</td>
<td>ILD, 4, 8, 16</td>
<td>ILD_1_1</td>
</tr>
<tr>
<td>ILDI_1_</td>
<td>ILDI_1</td>
<td>ILDI_1</td>
</tr>
<tr>
<td>ILDIX_1</td>
<td>ILDIX_1</td>
<td>ILDI_1_1</td>
</tr>
</tbody>
</table>
### Spartan-3 Design Elements

<table>
<thead>
<tr>
<th>Spartan-3 Design Elements</th>
<th>Spartan-3 Design Elements</th>
<th>Spartan-3 Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILDXI_1</td>
<td>INV, 4, 8, 16</td>
<td>IOBUF, IOBUF_selectIO</td>
</tr>
<tr>
<td>IOBUFDS</td>
<td>KEEPER</td>
<td>LD</td>
</tr>
<tr>
<td>LD_1</td>
<td>LD4, 8, 16</td>
<td>LDC</td>
</tr>
<tr>
<td>LDC_1</td>
<td>LDCE</td>
<td>LDCE_1</td>
</tr>
<tr>
<td>LD4CE, LD8CE, LD16CE</td>
<td>LDCP</td>
<td>LDCP_1</td>
</tr>
<tr>
<td>LDCPE</td>
<td>LDCPE_1</td>
<td>LDE</td>
</tr>
<tr>
<td>LDE_1</td>
<td>LDP</td>
<td>LDP_1</td>
</tr>
<tr>
<td>LDPE</td>
<td>LDPE_1</td>
<td>LUT1, 2, 3, 4</td>
</tr>
<tr>
<td>LUT1_D, LUT2_D, LUT3_D, LUT4_D</td>
<td>LUT1_L, LUT2_L, LUT3_L, LUT4_L</td>
<td>M2_1,</td>
</tr>
<tr>
<td>M2_1B1</td>
<td>M2_1B2</td>
<td>M2_1E</td>
</tr>
<tr>
<td>M4_1E</td>
<td>M8_1E</td>
<td>M16_1E</td>
</tr>
<tr>
<td>MULT_AND</td>
<td>MULT18X18</td>
<td>MULT18X18S</td>
</tr>
<tr>
<td>MUXCY, MUXF5, MUXF6, MUXF7</td>
<td>MUXCY_D, MUXF5_D, MUXF6_D</td>
<td>MUXF5_L, MUXF6_L, MUXF7_L</td>
</tr>
<tr>
<td>MUXF8</td>
<td>NAND2,9</td>
<td>NAND12, 16</td>
</tr>
<tr>
<td>NOR2-9</td>
<td>NOR12, 16</td>
<td>OBUF, 4, 8, 16</td>
</tr>
<tr>
<td>OBUF_selectIO</td>
<td>OBUFDS</td>
<td>OBUF, 4, 8, 16</td>
</tr>
<tr>
<td>OBUFUTDS</td>
<td>OFD, 4, 8, 16</td>
<td>OFD_1</td>
</tr>
<tr>
<td>OFDDRCE</td>
<td>OFDLE_1, OFDLE_2</td>
<td>OFDLE_1, OFDLE_2</td>
</tr>
<tr>
<td>OFDFTRSE</td>
<td>OFDLE_1, OFDLE_2</td>
<td>OFDLE_1, OFDLE_2</td>
</tr>
<tr>
<td>OFDI</td>
<td>OFDI_1</td>
<td>OFDT, 4, 8, 16</td>
</tr>
<tr>
<td>OFDT_1</td>
<td>OFDT, 4, 8, 16</td>
<td>OFDT_1</td>
</tr>
<tr>
<td>OFDI</td>
<td>OFDI_1</td>
<td>OFDI_1</td>
</tr>
<tr>
<td>OR2-9</td>
<td>OR12, 16</td>
<td>PULLDOWN</td>
</tr>
<tr>
<td>PULLUP</td>
<td>RAM16X1D</td>
<td>RAM16X1D_1</td>
</tr>
<tr>
<td>RAM16X1S</td>
<td>RAM16X1S_1</td>
<td>RAM16X1S</td>
</tr>
<tr>
<td>RAM16X4S</td>
<td>RAM16X8S</td>
<td>RAM32X1S</td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>RAM32X2S</td>
<td>RAM64X1S</td>
</tr>
<tr>
<td>RAM64X1S_1</td>
<td>RAMB16_Sn</td>
<td>RAMB16_Sm_Sn</td>
</tr>
<tr>
<td>ROC</td>
<td>ROCBUF</td>
<td>ROM16x1</td>
</tr>
<tr>
<td>ROM32X1</td>
<td>ROM64X1</td>
<td>ROM128X1</td>
</tr>
<tr>
<td>ROM256X1</td>
<td>SOP3-4</td>
<td>SR4CE, SR8CE, SR16CE</td>
</tr>
<tr>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>SR4CLED, SR8CLED, SR16CLED</td>
<td>SR4RE, SR8RE, SR16RE</td>
</tr>
<tr>
<td>SR4RE, SR8RE, SR16RE</td>
<td>SR4RELD, SR8RED, SR16REDL</td>
<td>SRL16</td>
</tr>
<tr>
<td>SRL16_1</td>
<td>SRL16E</td>
<td>SRL16_1</td>
</tr>
<tr>
<td>SRLC16</td>
<td>SRLC16_1</td>
<td>SRLC16E</td>
</tr>
<tr>
<td>SRLC16E_1</td>
<td>STARTBUF_architecture</td>
<td>STARTUP_SPARTAN3</td>
</tr>
<tr>
<td>TOC</td>
<td>TOCBUF</td>
<td>VCC</td>
</tr>
<tr>
<td>Spartan-3 Design Elements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XNOR2-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR2-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XORCY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XORCY_D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XORCY_L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The table below indicates the supported design elements for Virtex and Virtex-E. For a complete description, see the Product Data Sheets (http://www.xilinx.com/partinfo/databook.htm#virtex).

<table>
<thead>
<tr>
<th>Virtex, Virtex-E Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC4, 8, 16</td>
</tr>
<tr>
<td>AND2-9</td>
</tr>
<tr>
<td>BSCAN_VIRTEX</td>
</tr>
<tr>
<td>BUFE, 4, 8, 16</td>
</tr>
<tr>
<td>BUFGP</td>
</tr>
<tr>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
</tr>
<tr>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
</tr>
<tr>
<td>CC8CLED, CC16CLED</td>
</tr>
<tr>
<td>CD4CLE</td>
</tr>
<tr>
<td>CJ4CE, CJ5CE, CJ8CE</td>
</tr>
<tr>
<td>CLKDLLLE</td>
</tr>
<tr>
<td>COMPM2, 4, 8, 16</td>
</tr>
<tr>
<td>D2_4E</td>
</tr>
<tr>
<td>DEC_CC4, 8, 16</td>
</tr>
<tr>
<td>FD</td>
</tr>
<tr>
<td>FD4RE, FD8RE, FD16RE</td>
</tr>
<tr>
<td>FDCE</td>
</tr>
<tr>
<td>FDCP_1</td>
</tr>
<tr>
<td>FDE</td>
</tr>
<tr>
<td>FDP_1</td>
</tr>
<tr>
<td>FDR</td>
</tr>
<tr>
<td>FDRE_1</td>
</tr>
<tr>
<td>FDRSE</td>
</tr>
<tr>
<td>FDS_1</td>
</tr>
<tr>
<td>FJKC</td>
</tr>
<tr>
<td>FJKPE</td>
</tr>
<tr>
<td>FMAP</td>
</tr>
<tr>
<td>FTCLE</td>
</tr>
<tr>
<td>FTP</td>
</tr>
<tr>
<td>FTRSELE</td>
</tr>
<tr>
<td>GND</td>
</tr>
<tr>
<td>IBUFEG, IBUFG_selectIO</td>
</tr>
<tr>
<td>IFDI</td>
</tr>
<tr>
<td>IFDX_1</td>
</tr>
<tr>
<td>ILD, 4, 8, 16</td>
</tr>
<tr>
<td>ILDI_1</td>
</tr>
</tbody>
</table>
## Virtex, Virtex-E Design Elements

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILDXI</td>
<td>ILDXI_1</td>
<td>INV, 4, 8, 16</td>
</tr>
<tr>
<td>IOBUF, IOBUF_selectIO</td>
<td>IOPAD, 4, 8, 16</td>
<td>IPAD, 4, 8, 16</td>
</tr>
<tr>
<td>KEEPER</td>
<td>LD</td>
<td>LD_1</td>
</tr>
<tr>
<td>LD4, 8, 16</td>
<td>LDC</td>
<td>LDC_1</td>
</tr>
<tr>
<td>LDCE</td>
<td>LDCE_1</td>
<td>LD4CE, LD8CE, LD16CE</td>
</tr>
<tr>
<td>LDCP</td>
<td>LDCP_1</td>
<td>LDCPE</td>
</tr>
<tr>
<td>LDCPE_1</td>
<td>LDE</td>
<td>LDE_1</td>
</tr>
<tr>
<td>LDPE</td>
<td>LDPE_1</td>
<td></td>
</tr>
<tr>
<td>LD1</td>
<td>LUT1, 2, 3, 4</td>
<td>LUT1_D, LUT2_D, LUT3_D, LUT4_D</td>
</tr>
<tr>
<td>LUT1_1, LUT2_L, LUT3_L, LUT4_L</td>
<td>M2_1</td>
<td>M2_1BI</td>
</tr>
<tr>
<td>M2_1B2</td>
<td>M2_1E</td>
<td>M4_1E</td>
</tr>
<tr>
<td>M8_1E</td>
<td>M16_1E</td>
<td>MULT_AND</td>
</tr>
<tr>
<td>MUXCY</td>
<td>MUXCY_D</td>
<td>MUXCY_L</td>
</tr>
<tr>
<td>MUXF5</td>
<td>MUXF5_D</td>
<td>MUXF5_L</td>
</tr>
<tr>
<td>MUXF6</td>
<td>MUXF6_D</td>
<td>MUXF6_L</td>
</tr>
<tr>
<td>NAND2-9</td>
<td>NAND12, 16</td>
<td>NOR2-9</td>
</tr>
<tr>
<td>NOR12, 16</td>
<td>OBUF, 4, 8, 16</td>
<td>OBUF_selectIO</td>
</tr>
<tr>
<td>OBUF, 4, 8, 16</td>
<td>OFD, 4, 8, 16</td>
<td>OFDE, 4, 8, 16</td>
</tr>
<tr>
<td>OFD_1</td>
<td>OFD</td>
<td>OFDI_1</td>
</tr>
<tr>
<td>OFDE_1</td>
<td>OFDI</td>
<td>OFDX, 4, 8, 16</td>
</tr>
<tr>
<td>OFD</td>
<td>OFD_1</td>
<td>OFDX, 4, 8, 16</td>
</tr>
<tr>
<td>OFDT_1</td>
<td>OFDT</td>
<td>OFDI_1</td>
</tr>
<tr>
<td>OFDX_1</td>
<td>OFDXI</td>
<td>OFDI_1</td>
</tr>
<tr>
<td>ORAD, 4, 8, 16</td>
<td>OR2-9</td>
<td>OR12, 16</td>
</tr>
<tr>
<td>PULLDOWN</td>
<td>PULLUP</td>
<td>RAM16X1D</td>
</tr>
<tr>
<td>RAM16X1D_1</td>
<td>RAM16X1S</td>
<td>RAM16X1S_1</td>
</tr>
<tr>
<td>RAM16X2D</td>
<td>RAM16X2S</td>
<td>RAM16X4D</td>
</tr>
<tr>
<td>RAM16X4S</td>
<td>RAM16X8D</td>
<td>RAM16X8S</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>RAM32X1S_1</td>
<td>RAM32X2S</td>
</tr>
<tr>
<td>RAM32X4S</td>
<td>RAM32X8S</td>
<td>RAMB4_Sn</td>
</tr>
<tr>
<td>RAMB4_Sm_Sn</td>
<td>ROC</td>
<td>ROCBUF</td>
</tr>
<tr>
<td>ROM16X1</td>
<td>ROM32X1</td>
<td>SOP3-4</td>
</tr>
<tr>
<td>SR4CE, SR8CE, SR16CE</td>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>SR4CLED, SR8CLED, SR16CLED</td>
</tr>
<tr>
<td>SR4RE, SR8RE, SR16RE</td>
<td>SR4RLE, SR8RLE, SR16RLE</td>
<td>SR4RLED, SR8RLED, SR16RLED</td>
</tr>
<tr>
<td>SRL16</td>
<td>SRL16_1</td>
<td>SRL16E</td>
</tr>
<tr>
<td>SRL16E_1</td>
<td>STARTBUF_architecture</td>
<td>STARTUP_VIRTTEX</td>
</tr>
<tr>
<td>TOC</td>
<td>TOCBUF</td>
<td>UPAD</td>
</tr>
<tr>
<td>VCC</td>
<td>XNOR2-9</td>
<td>XOR2-9</td>
</tr>
<tr>
<td>XORCY</td>
<td>XORCY_D</td>
<td>XORCY_L</td>
</tr>
</tbody>
</table>
The table below indicates the supported design elements for Virtex-II, Virtex-II Pro, and Virtex-II Pro X. For a complete description of Virtex-II, see the Product Data Sheets (http://www.xilinx.com/partinfo/databook.htm#vtwo).

### Virtex-II, Virtex-II Pro, Virtex-II Pro X Design Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Virtex-II</th>
<th>Virtex-II Pro</th>
<th>Virtex-II Pro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC4, 8, 16</td>
<td>ADD4, 8, 16</td>
<td>ADSU4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>AND2-9</td>
<td>AND12, 16</td>
<td>BRLSHT4, 8</td>
<td></td>
</tr>
<tr>
<td>BSCAN_VIRTEX2</td>
<td>BUF</td>
<td>BUFCF</td>
<td></td>
</tr>
<tr>
<td>BUFE, 4, 8, 16</td>
<td>BUFG</td>
<td>BUFGCE</td>
<td></td>
</tr>
<tr>
<td>BUFGCE_1</td>
<td>BUFGDLL</td>
<td>BUFGMUX</td>
<td></td>
</tr>
<tr>
<td>BUFGMUX_1</td>
<td>BUF</td>
<td>BUFT, 8, 16</td>
<td></td>
</tr>
<tr>
<td>CAPTURE_VIRTEX2</td>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
<td></td>
</tr>
<tr>
<td>CB2CLED, CB4CLED, CB8CLED, CB16CLED</td>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
<td>CC8CE, CC16CE</td>
<td></td>
</tr>
<tr>
<td>CC8CLE, CC16CLE</td>
<td>CC8CLED, CC16CLELED</td>
<td>CC8RE, CC16RE</td>
<td></td>
</tr>
<tr>
<td>CD4CE</td>
<td>CD4CLE</td>
<td>CD4RE</td>
<td></td>
</tr>
<tr>
<td>CD4RLE</td>
<td>CJ4CE, CJ5CE, CJ8CE</td>
<td>CJ4RE, CJ5RE, CJ8RE</td>
<td></td>
</tr>
<tr>
<td>COMP2, 4, 8, 16</td>
<td>COMPM2, 4, 8, 16</td>
<td>COMPMC8, 16</td>
<td></td>
</tr>
<tr>
<td>CR8CE, CR16CE</td>
<td>D2_4E</td>
<td>D3_8E</td>
<td></td>
</tr>
<tr>
<td>D4_16E</td>
<td>DCM</td>
<td>DEC_CC, 8, 16</td>
<td></td>
</tr>
<tr>
<td>DECODE4, 8, 16</td>
<td>DECODE32, 64</td>
<td>FD</td>
<td></td>
</tr>
<tr>
<td>FD_1</td>
<td>FD4CE, FD8CE, FD16CE</td>
<td>FD4RE, FD8RE, FD16RE</td>
<td></td>
</tr>
<tr>
<td>FDC</td>
<td>FDC_1</td>
<td>FDCE</td>
<td></td>
</tr>
<tr>
<td>FDCE_1</td>
<td>FDCP</td>
<td>FDCP_1</td>
<td></td>
</tr>
<tr>
<td>FDCP</td>
<td>FDCP_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDCP</td>
<td>FDCP_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDDRSE</td>
<td>FDE</td>
<td>FDE_1</td>
<td></td>
</tr>
<tr>
<td>FDP_1</td>
<td>FDP</td>
<td>FDP_1</td>
<td></td>
</tr>
<tr>
<td>FDP</td>
<td>FDP</td>
<td>FDP_1</td>
<td></td>
</tr>
<tr>
<td>FDRE</td>
<td>FDR</td>
<td>FDR_1</td>
<td></td>
</tr>
<tr>
<td>FDR_1</td>
<td>FDRS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDRS</td>
<td>FDRSE_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDRS</td>
<td>FDRSE_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDS_1</td>
<td>FDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDS</td>
<td>FDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDSE_1</td>
<td>FJKC</td>
<td>FJKCE</td>
<td></td>
</tr>
<tr>
<td>FJKP</td>
<td>FJKP</td>
<td>FJKRSE</td>
<td></td>
</tr>
<tr>
<td>FJKSRE</td>
<td>FMAP</td>
<td>FTC</td>
<td></td>
</tr>
<tr>
<td>FTCE</td>
<td>FTCLE</td>
<td>FTCLEX</td>
<td></td>
</tr>
<tr>
<td>FTP</td>
<td>FTP</td>
<td>FTPLE</td>
<td></td>
</tr>
<tr>
<td>FTP</td>
<td>FTP</td>
<td>FTPLE</td>
<td></td>
</tr>
<tr>
<td>FTRSE</td>
<td>FTRSLE</td>
<td>FTSRE</td>
<td></td>
</tr>
<tr>
<td>FTSRLE</td>
<td>GND</td>
<td>GT_AURORA_n (Virtex-II Pro only)</td>
<td></td>
</tr>
<tr>
<td>GT_CUSTOM (Virtex-II Pro only)</td>
<td>GT_ETHERNET_n (Virtex-II Pro only)</td>
<td>GT_FIBRE_CHAN_n (Virtex-II Pro only)</td>
<td></td>
</tr>
<tr>
<td>GT_INFINIBAND_n (Virtex-II Pro only)</td>
<td>GT_XAUI_n (Virtex-II Pro only)</td>
<td>GT10_AURORA_n (Virtex-II Pro X only)</td>
<td></td>
</tr>
<tr>
<td>GT10_AURORAX_n (Virtex-II Pro X only)</td>
<td>GT10_CUSTOM (Virtex-II Pro X only)</td>
<td>GT10_INFINIBAND_n (Virtex-II Pro X only)</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X Design Elements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT10_XAUI_n (Virtex-II Pro X only)</td>
<td>GT10_10GE_n (Virtex-II Pro X only)</td>
<td>GT10_10GFC_n (Virtex-II Pro X only)</td>
<td></td>
</tr>
<tr>
<td>GT10_OC48_n (Virtex-II Pro X only)</td>
<td>GT10_OC192_n (Virtex-II Pro X only)</td>
<td>IBUF, 4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>IBUF_selectIO</td>
<td>IBUFDS</td>
<td>IBUFG, IBUFG_selectIO</td>
<td></td>
</tr>
<tr>
<td>IBUFGDS</td>
<td>IBUFGDS_DIFF_OUT</td>
<td>ICAP_VIRTEX2</td>
<td></td>
</tr>
<tr>
<td>IFD, 4, 8, 16</td>
<td>IFD_1</td>
<td>IFDDRCPE</td>
<td></td>
</tr>
<tr>
<td>IFDDRRSE</td>
<td>IFDI</td>
<td>IFDI_1</td>
<td></td>
</tr>
<tr>
<td>IFDX, 4, 8, 16</td>
<td>IFDX_1</td>
<td>IFDXI</td>
<td></td>
</tr>
<tr>
<td>IFDIXI_1</td>
<td>ILD, 4, 8, 16</td>
<td>ILD_1</td>
<td></td>
</tr>
<tr>
<td>ILD1</td>
<td>ILD1_1</td>
<td>ILDX, 4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>ILDX1</td>
<td>ILDX1_1</td>
<td>ILDX1</td>
<td></td>
</tr>
<tr>
<td>INV, 4, 8, 16</td>
<td>IOBUF, IOBUF_selectIO</td>
<td>IOBUFDS</td>
<td></td>
</tr>
<tr>
<td>IOPAD, 4, 8, 16</td>
<td>IPAD, 4, 8, 16</td>
<td>JTAGPPC (Virtex-II Pro only)</td>
<td></td>
</tr>
<tr>
<td>KEEPER</td>
<td>LD</td>
<td>LD_1</td>
<td></td>
</tr>
<tr>
<td>LD4, 8, 16</td>
<td>LDC</td>
<td>LDC_1</td>
<td></td>
</tr>
<tr>
<td>LDC</td>
<td>LDC1_1</td>
<td>LD4CE, LD8CE, LD16CE</td>
<td></td>
</tr>
<tr>
<td>LDCP</td>
<td>LDCP_1</td>
<td>LDCPE</td>
<td></td>
</tr>
<tr>
<td>LDCP1_1</td>
<td>LDE</td>
<td>LDE_1</td>
<td></td>
</tr>
<tr>
<td>LDP</td>
<td>LDP1_1</td>
<td>LDPE</td>
<td></td>
</tr>
<tr>
<td>LDPE1_1</td>
<td>LUT1, 2, 3, 4</td>
<td>LUT1_D, LUT2_D, LUT3_D, LUT4_D</td>
<td></td>
</tr>
<tr>
<td>LUT1_L, LUT2_L, LUT3_L, LUT4_L</td>
<td>M2_1</td>
<td>M2_1B1</td>
<td></td>
</tr>
<tr>
<td>M2_1B2</td>
<td>M2_1E</td>
<td>M4_1E</td>
<td></td>
</tr>
<tr>
<td>M8_1E</td>
<td>M16_1E</td>
<td>MULT_AND</td>
<td></td>
</tr>
<tr>
<td>MULT18X18</td>
<td>MULT18X18S</td>
<td>MUXCY</td>
<td></td>
</tr>
<tr>
<td>MUXCY_D</td>
<td>MUXCY_L</td>
<td>MUXF5</td>
<td></td>
</tr>
<tr>
<td>MUXF5_D</td>
<td>MUXF5_L</td>
<td>MUXF6</td>
<td></td>
</tr>
<tr>
<td>MUXF6_D</td>
<td>MUXF6_L</td>
<td>MUXF7</td>
<td></td>
</tr>
<tr>
<td>MUXF7_D</td>
<td>MUXF7_L</td>
<td>MUXF8</td>
<td></td>
</tr>
<tr>
<td>NAND2-9</td>
<td>NAND12, 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOR2-9</td>
<td>NOR12, 16</td>
<td>OBUF, 4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>OBUF_selectIO</td>
<td>OBUFDS</td>
<td>OBUFE, 4, 8, 16</td>
<td></td>
</tr>
<tr>
<td>OBUFT, 4, 8, 16</td>
<td>OBUFT_selectIO</td>
<td>OBUFTDS</td>
<td></td>
</tr>
<tr>
<td>OFD, 4, 8, 16</td>
<td>OFD_1</td>
<td>OFDDRCPE</td>
<td></td>
</tr>
<tr>
<td>OFDDRRSE</td>
<td>OFDDRTCPE</td>
<td>OFDDRTRSE</td>
<td></td>
</tr>
<tr>
<td>OFDE, 4, 8, 16</td>
<td>OFDE_1</td>
<td>OFDI</td>
<td></td>
</tr>
<tr>
<td>OFDI_1</td>
<td>OFDT, 4, 8, 16</td>
<td>OFDT_1</td>
<td></td>
</tr>
<tr>
<td>OFDX, 4, 8, 16</td>
<td>OFDX_1</td>
<td>OFDXI</td>
<td></td>
</tr>
<tr>
<td>OFDXX1_1</td>
<td>OPAD, 4, 8, 16</td>
<td>OR2-9</td>
<td></td>
</tr>
<tr>
<td>OR12, 16</td>
<td>ORCY</td>
<td>PPC405 (Virtex-II Pro only)</td>
<td></td>
</tr>
<tr>
<td>PULLDOWN</td>
<td>PULLUP</td>
<td>RAM16X1D</td>
<td></td>
</tr>
<tr>
<td>RAM16X1D_1</td>
<td>RAM16X1S</td>
<td>RAM16X1S_1</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Component</td>
<td>Component</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td>RAM16X2D</td>
<td>RAM16X2S</td>
<td>RAM16X4D</td>
<td></td>
</tr>
<tr>
<td>RAM16X4S</td>
<td>RAM16X8D</td>
<td>RAM16X8S</td>
<td></td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>RAM32X1D_1</td>
<td>RAM32X1S</td>
<td></td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>RAM32X2S</td>
<td>RAM32X4S</td>
<td></td>
</tr>
<tr>
<td>RAM32X8S</td>
<td>RAM64X1D</td>
<td>RAM64X1D_1</td>
<td></td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>RAM64X1S_1</td>
<td>RAM64X2S</td>
<td></td>
</tr>
<tr>
<td>RAM128X1S</td>
<td>RAM128X1S_1</td>
<td>RAMB16_Sn</td>
<td></td>
</tr>
<tr>
<td>RAMB16_Sm_Sn</td>
<td>ROC</td>
<td>ROCBUF</td>
<td></td>
</tr>
<tr>
<td>ROM16X1</td>
<td>ROM32X1</td>
<td>ROM64X1</td>
<td></td>
</tr>
<tr>
<td>ROMI256X1</td>
<td>ROM256X1</td>
<td>SOP3-4</td>
<td></td>
</tr>
<tr>
<td>SR4CE, SR8CE, SR16CE</td>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>SR4CLED, SR8CLED, SR16CLED</td>
<td></td>
</tr>
<tr>
<td>SR4RE, SR8RE, SR16RE</td>
<td>SR4RLE, SR8RLE, SR16RLE</td>
<td>SR4RLED, SR8RLED, SR16RLED</td>
<td></td>
</tr>
<tr>
<td>SRL16</td>
<td>SRL16_1</td>
<td>SRL16E</td>
<td></td>
</tr>
<tr>
<td>SRL16E_1</td>
<td>SRLC16</td>
<td>SRLC16_1</td>
<td></td>
</tr>
<tr>
<td>SRLC16E</td>
<td>SRLC16E_1</td>
<td>STARTBUF_architecture</td>
<td></td>
</tr>
<tr>
<td>STARTUP_VIRTEX2</td>
<td>TOC</td>
<td>TOCBUF</td>
<td></td>
</tr>
<tr>
<td>UPAD</td>
<td>VCC</td>
<td>XNOR2-9</td>
<td></td>
</tr>
<tr>
<td>XOR2-9</td>
<td>XORCY</td>
<td>XORCY_D</td>
<td></td>
</tr>
<tr>
<td>XORCY_L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The table below indicates the supported design elements for XC95000/XV/XL. For a complete description, see the Product Data Sheets (http://www.xilinx.com/partinfo/databook.htm#cpld).

**XC9500/XV/XL Design Elements**

<table>
<thead>
<tr>
<th>ACC1</th>
<th>ACC4, 8, 16</th>
<th>ADD1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD4, 8, 16</td>
<td>ADSU1</td>
<td>ADSU4, 8, 16</td>
</tr>
<tr>
<td>AND2-9</td>
<td>BRLSHT4, 8</td>
<td>BUF</td>
</tr>
<tr>
<td>BUF4, 8, 16</td>
<td>BUFE, 4, 8, 16</td>
<td>BUFG</td>
</tr>
<tr>
<td>BUFGRS</td>
<td>BUFGTS</td>
<td>BUFT, 4, 8, 16</td>
</tr>
<tr>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
<td>CB2CLED, CB4CLED, CB8CLED, CB16CLED</td>
</tr>
<tr>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
</tr>
<tr>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CD4CE</td>
<td>CD4CLE</td>
</tr>
<tr>
<td>CD4RE</td>
<td>CD4RLE</td>
<td>CJ4CE, CJ5CE, CJ8CE</td>
</tr>
<tr>
<td>CJ4RE, CJ5RE, CJ8RE</td>
<td>COMP2, 4, 8, 16</td>
<td>COMPM2, 4, 8, 16</td>
</tr>
<tr>
<td>CR8CE, CR16CE</td>
<td>D2_4E</td>
<td>D3_8E</td>
</tr>
<tr>
<td>D4_16E</td>
<td>FD</td>
<td>FD4, 8, 16</td>
</tr>
<tr>
<td>FD4CE, FD8CE, FD16CE</td>
<td>FD4RE, FD8RE, FD16RE</td>
<td>FDC</td>
</tr>
<tr>
<td>FDCE</td>
<td>FDCP</td>
<td>FDCPE</td>
</tr>
<tr>
<td>FDP</td>
<td>FDPE</td>
<td>FDR</td>
</tr>
<tr>
<td>FDR</td>
<td>FDRS</td>
<td>FDRSE</td>
</tr>
<tr>
<td>FDS</td>
<td>FDSE</td>
<td>FDSR</td>
</tr>
<tr>
<td>FDSR</td>
<td>FJKC</td>
<td>FJKCE</td>
</tr>
<tr>
<td>FJKCP</td>
<td>FJKCPE</td>
<td>FJKP</td>
</tr>
<tr>
<td>FJKPE</td>
<td>FJKRSE</td>
<td>FJKSRE</td>
</tr>
<tr>
<td>FTC</td>
<td>FTCCE</td>
<td>FT克莱</td>
</tr>
<tr>
<td>FTCP</td>
<td>FTCPE</td>
<td>FTCPE</td>
</tr>
<tr>
<td>FTCDCP</td>
<td>FTP</td>
<td>FTP</td>
</tr>
<tr>
<td>FTP</td>
<td>FTPRSE</td>
<td>FTPRSE</td>
</tr>
<tr>
<td>FTPRE</td>
<td>FTPRSE</td>
<td>FTPRSE</td>
</tr>
<tr>
<td>FTSRE</td>
<td>FTSRLE</td>
<td>GND</td>
</tr>
<tr>
<td>IBUF, 4, 8, 16</td>
<td>INV, 4, 8, 16</td>
<td>IOPAD, 4, 8, 16</td>
</tr>
<tr>
<td>IPAD, 4, 8, 16</td>
<td>LD</td>
<td>LD4, 8, 16</td>
</tr>
<tr>
<td>LDC</td>
<td>LDCP</td>
<td>LDP</td>
</tr>
<tr>
<td>M2_1</td>
<td>M2_1B1</td>
<td>M2_1B2</td>
</tr>
<tr>
<td>M2_1E</td>
<td>M4_1E</td>
<td>M8_1E</td>
</tr>
<tr>
<td>M16_1E</td>
<td>NAND2-9</td>
<td>NOR2-9</td>
</tr>
<tr>
<td>OBUF, 4, 8, 16</td>
<td>OBUFE, 4, 8, 16</td>
<td>OBUFT, 4, 8, 16</td>
</tr>
<tr>
<td>OPAD, 4, 8, 16</td>
<td>OR2-9</td>
<td>SOP3-4</td>
</tr>
<tr>
<td>SR4CE, SR8CE, SR16CE</td>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>SR4CLED, SR8CLED, SR16CLED</td>
</tr>
<tr>
<td>SR4RE, SR8RE, SR16RE</td>
<td>SR4RLE, SR8RLE, SR16RLE</td>
<td>SR4RLED, SR8RLED, SR16RLED</td>
</tr>
<tr>
<td>VCC</td>
<td>XNOR2-9</td>
<td>XOR2-9</td>
</tr>
</tbody>
</table>
The table below indicates the supported design elements for CoolRunner XPLA3. For a complete description, see the Product Data Sheets (http://www.xilinx.com/partinfo/databook.htm#coolrunner).

**CoolRunner XPLA3 Design Elements**

<table>
<thead>
<tr>
<th>Design Elements</th>
<th>Design Elements</th>
<th>Design Elements</th>
<th>Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC1</td>
<td>ACC4, 8, 16</td>
<td>ACC4, 8, 16</td>
<td>ADD1</td>
</tr>
<tr>
<td>ADD4, 8, 16</td>
<td>ADSU1</td>
<td>ADSU1, 4, 8, 16</td>
<td>ADD1</td>
</tr>
<tr>
<td>AND2-9</td>
<td>BRLSHFT4, 8</td>
<td>BRLSHFT4, 8</td>
<td>BUFE, 4, 8, 16</td>
</tr>
<tr>
<td>BUFE, 4, 8, 16</td>
<td>BUFE, 4, 8, 16</td>
<td>BUFE, 4, 8, 16</td>
<td>BUFG</td>
</tr>
<tr>
<td>BUFT, 4, 8, 16</td>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
<td></td>
</tr>
<tr>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CD4CE</td>
</tr>
<tr>
<td>CD4CLE</td>
<td>CD4RE</td>
<td>CD4RE</td>
<td>CD4RE</td>
</tr>
<tr>
<td>CJ4CE, CJ5CE, CJ8CE</td>
<td>CJ4RE, CJ5RE, CJ8RE</td>
<td>CJ4RE, CJ5RE, CJ8RE</td>
<td>COMP2, 4, 8, 16</td>
</tr>
<tr>
<td>COMP2, 4, 8, 16</td>
<td>CR8CE, CR16CE</td>
<td>D2_4E</td>
<td>FD4, 8, 16</td>
</tr>
<tr>
<td>D3_8E</td>
<td>D4_16E</td>
<td>D4_16E</td>
<td>D4_16E</td>
</tr>
<tr>
<td>FDC</td>
<td>FDCE</td>
<td>FDCE</td>
<td>FDCE</td>
</tr>
<tr>
<td>FDCPE</td>
<td>FDP</td>
<td>FDP</td>
<td>FDP</td>
</tr>
<tr>
<td>FDR</td>
<td>FDRE</td>
<td>FDRE</td>
<td>FDRE</td>
</tr>
<tr>
<td>FDRSE</td>
<td>FDS</td>
<td>FDS</td>
<td>FDS</td>
</tr>
<tr>
<td>FDSR</td>
<td>FDSRE</td>
<td>FDSRE</td>
<td>FDSRE</td>
</tr>
<tr>
<td>FJKC</td>
<td>FJKCP</td>
<td>FJKCP</td>
<td>FJKCP</td>
</tr>
<tr>
<td>FJKP</td>
<td>FJKPE</td>
<td>FJKPE</td>
<td>FJKPE</td>
</tr>
<tr>
<td>FJKP</td>
<td>FJKRSE</td>
<td>FJKRSE</td>
<td>FJKRSE</td>
</tr>
<tr>
<td>FJKSRE</td>
<td>FTC</td>
<td>FTC</td>
<td>FTC</td>
</tr>
<tr>
<td>FTCLE</td>
<td>FTC</td>
<td>FTC</td>
<td>FTC</td>
</tr>
<tr>
<td>FTCLE</td>
<td>FTC</td>
<td>FTC</td>
<td>FTC</td>
</tr>
<tr>
<td>FTCLE</td>
<td>FTC</td>
<td>FTC</td>
<td>FTC</td>
</tr>
<tr>
<td>FTCLE</td>
<td>FTC</td>
<td>FTC</td>
<td>FTC</td>
</tr>
<tr>
<td>FTDCP</td>
<td>FTP</td>
<td>FTP</td>
<td>FTP</td>
</tr>
<tr>
<td>FTP</td>
<td>FTP</td>
<td>FTP</td>
<td>FTP</td>
</tr>
<tr>
<td>FTRSE</td>
<td>FTSRE</td>
<td>FTSRE</td>
<td>FTSRE</td>
</tr>
<tr>
<td>FTRSE</td>
<td>FTSRE</td>
<td>FTSRE</td>
<td>FTSRE</td>
</tr>
<tr>
<td>GND</td>
<td>IBUF, 4, 8, 16</td>
<td>INV, 4, 8, 16</td>
<td>LD</td>
</tr>
<tr>
<td>IOPAD, 4, 8, 16</td>
<td>IPAD, 4, 8, 16</td>
<td>IPAD, 4, 8, 16</td>
<td>LD</td>
</tr>
<tr>
<td>LD4, 8, 16</td>
<td>LDC</td>
<td>LDC</td>
<td>LDC</td>
</tr>
<tr>
<td>LD4</td>
<td>LDC</td>
<td>LDC</td>
<td>LDC</td>
</tr>
<tr>
<td>LDP</td>
<td>M2_1</td>
<td>M2_1</td>
<td>M2_1</td>
</tr>
<tr>
<td>M2_1B2</td>
<td>M2_1E</td>
<td>M2_1E</td>
<td>M2_1E</td>
</tr>
<tr>
<td>M8_1E</td>
<td>M8_1E</td>
<td>M8_1E</td>
<td>M8_1E</td>
</tr>
<tr>
<td>M8_1E</td>
<td>M8_1E</td>
<td>M8_1E</td>
<td>M8_1E</td>
</tr>
<tr>
<td>M8_1E</td>
<td>M8_1E</td>
<td>M8_1E</td>
<td>M8_1E</td>
</tr>
<tr>
<td>NOR2-9</td>
<td>OBUF, 4, 8, 16</td>
<td>OBUF, 4, 8, 16</td>
<td>OBUF, 4, 8, 16</td>
</tr>
<tr>
<td>OBUF4, 4, 8, 16</td>
<td>OPAD, 4, 8, 16</td>
<td>OPAD, 4, 8, 16</td>
<td>OR2-9</td>
</tr>
<tr>
<td>OR2-9</td>
<td>SR4CE, SR8CE, SR16CE</td>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td></td>
</tr>
<tr>
<td>SR4RLED, SR8RLED, SR16RLED</td>
<td>VCC</td>
<td>VCC</td>
<td>XNOR2-9</td>
</tr>
<tr>
<td>XNOR2-9</td>
<td>XNOR2-9</td>
<td>XNOR2-9</td>
<td>XNOR2-9</td>
</tr>
</tbody>
</table>

Libraries Guide
ISE 6.1
www.xilinx.com
1-800-255-7778
53
The table below indicates the supported design elements for CoolRunner-II. For a complete description of CoolRunner-II, see the Product Data Sheets (http://www.xilinx.com/partinfo/databook.htm#cooltwo).

### CoolRunner-II Design Elements

<table>
<thead>
<tr>
<th>ACC1</th>
<th>ACC4, 8, 16</th>
<th>ADD1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD4, 8, 16</td>
<td>ADSU1</td>
<td>ADSU4, 8, 16</td>
</tr>
<tr>
<td>AND2-9</td>
<td>BRLSHFT4, 8</td>
<td>BUF</td>
</tr>
<tr>
<td>BUF4, 8, 16</td>
<td>BUF</td>
<td>BUFGSR</td>
</tr>
<tr>
<td>BUFGTS</td>
<td>CB2CE, CB4CE, CB8CE, CB16CE</td>
<td>CB2CLE, CB4CLE, CB8CLE, CB16CLE</td>
</tr>
<tr>
<td>CB2CLED, CB4CLED, CB8CLED, CB16CLED</td>
<td>CB2RE, CB4RE, CB8RE, CB16RE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
</tr>
<tr>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td></td>
</tr>
<tr>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
</tr>
<tr>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td></td>
</tr>
<tr>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
</tr>
<tr>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td></td>
</tr>
<tr>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
</tr>
<tr>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td></td>
</tr>
<tr>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
</tr>
<tr>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td></td>
</tr>
<tr>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
<tr>
<td>CB2X1, CB4X1, CB8X1, CB16X1</td>
<td>CB2X2, CB4X2, CB8X2, CB16X2</td>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
</tr>
<tr>
<td>CB2DCE, CB4DCE, CB8DCE, CB16DCE</td>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td></td>
</tr>
<tr>
<td>CB2DRE, CB4DRE, CB8DRE, CB16DRE</td>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td></td>
</tr>
</tbody>
</table>

### Libraries Guide

ISE 6.1i  
www.xilinx.com  
1-800-255-7778
## CoolRunner-II Design Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTDRSLE</td>
<td>FTP</td>
<td>FTPE</td>
</tr>
<tr>
<td>FTPLE</td>
<td>FTRSE</td>
<td>FTRSLE</td>
</tr>
<tr>
<td>FTSRE</td>
<td>FTSRLE</td>
<td>GND</td>
</tr>
<tr>
<td>IBUF, 4, 8, 16</td>
<td>INV, 4, 8, 16</td>
<td>IOPAD, 4, 8, 16</td>
</tr>
<tr>
<td>IPAD, 4, 8, 16</td>
<td>KEEPER</td>
<td>LD</td>
</tr>
<tr>
<td>LD4, 8, 16</td>
<td>LDC</td>
<td>LDCP</td>
</tr>
<tr>
<td>LDG</td>
<td>LDG4, 8, 16</td>
<td>LDP</td>
</tr>
<tr>
<td>M2_1</td>
<td>M2_1B1</td>
<td>M2_1B2</td>
</tr>
<tr>
<td>M2_1E</td>
<td>M4_1E</td>
<td>M8_1E</td>
</tr>
<tr>
<td>M16_1E</td>
<td>NAND2-9</td>
<td>NOR2-9</td>
</tr>
<tr>
<td>OBUF, 4, 8, 16</td>
<td>OBUFE, 4, 8, 16</td>
<td>OBUFT, 4, 8, 16</td>
</tr>
<tr>
<td>OPAD, 4, 8, 16</td>
<td>OR2-9</td>
<td>PULLDOWN</td>
</tr>
<tr>
<td>PULLUP</td>
<td>SOP3-4</td>
<td>SR4CE, SR8CE, SR16CE</td>
</tr>
<tr>
<td>SR4CLE, SR8CLE, SR16CLE</td>
<td>SR4CLED, SR8CLED, SR16CLED</td>
<td>SR4RE, SR8RE, SR16RE</td>
</tr>
<tr>
<td>SR4RLE, SR8RLE, SR16RLE</td>
<td>SR4RLED, SR8RLED, SR16RLED</td>
<td>SRD4CE, SRD8CE, SRD16CE</td>
</tr>
<tr>
<td>SRD4CLE, SRD8CLE, SRD16CLE</td>
<td>SRD4CLED, SRD8CLED, SRD16CLED</td>
<td>SRD4RE, SRD8RE, SRD16RE</td>
</tr>
<tr>
<td>SRD4RLE, SRD8RLE, SRD16RLE</td>
<td>SRD4RLED, SRD8RLED, SRD16RLED</td>
<td>VCC</td>
</tr>
<tr>
<td>XNOR2-9</td>
<td>XOR2-9</td>
<td></td>
</tr>
</tbody>
</table>
Functional Categories

This section categorizes, by function, the logic elements that are described in detail in the “Design Elements” sections. Each category is briefly described. Tables under each category identify all the available elements for the function and indicate which architectures are supported by each.

<table>
<thead>
<tr>
<th>Arithmetic Functions</th>
<th>Flip-Flops</th>
<th>Logic Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffers</td>
<td>General</td>
<td>Map Elements</td>
</tr>
<tr>
<td>Comparators</td>
<td>Input Latches</td>
<td>Memory Elements</td>
</tr>
<tr>
<td>Counters</td>
<td>Input/Output Flip-Flops</td>
<td>Multiplexers</td>
</tr>
<tr>
<td>Decoders</td>
<td>Input/Output Functions</td>
<td>Shifters</td>
</tr>
<tr>
<td>Edge Decoders</td>
<td>Latches</td>
<td>Shift Registers</td>
</tr>
</tbody>
</table>

Elements are listed in alphanumeric order under each category.

See the Xilinx Unified Libraries chapter for information on the specific device families that use each library. "No" column means that the element does not apply.

The Xilinx libraries contain three types of elements.

- Primitives are basic logical elements such as AND2 and OR2 gates.
- Soft macros are schematics made by combining primitives and sometimes other soft macros.
- Relationally placed macros (RPMs) are soft macros that contain relative location constraint (RLOC) information, carry logic symbols, and FMAP symbols, where appropriate.

The last item mentioned above, RPMs, applies only to FPGA families.

The relationally placed macro (RPM) library uses RLOC constraints to define the order and structure of the underlying design primitives. Because these macros are built upon standard schematic parts, they do not have to be translated before simulation. The components that are implemented as RPMs are listed in the “Slice Count” section.

Designs created with RPMs can be functionally simulated. RPMs can, but need not, include all the following elements.

- FMAPs and CLB-grouping attributes to control mapping. FMAPs have pin-lock attributes, which allow better control over routing.
- Relative location (RLOC) constraints to provide placement structure. They allow positioning of elements relative to each other.
- Carry logic primitive symbols.
The RPM library offers the functionality and precision of the hard macro library with added flexibility. You can optimize RPMs and merge other logic within them. The elements in the RPM library allow you to access carry logic easily and to control mapping and block placement. Because RPMs are a superset of ordinary macros, you can design them in the normal design entry environment. They can include any primitive logic. The macro logic is fully visible to you and can be easily back-annotated with timing information.
Arithmetic Functions

There are three types of arithmetic functions: accumulators (ACC), adders (ADD), and adder/subtracters (ADSU). With an ADSU, either unsigned binary or twos-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated. The following figure shows the ADSU carry-out and overflow boundaries.

### ADSU Carry-Out and Overflow Boundaries

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC1</td>
<td>1-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ACC4</td>
<td>4-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ACC8</td>
<td>8-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ACC16</td>
<td>16-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADD1</td>
<td>1-Bit Full Adder with Carry-In and Carry-Out</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADD4</td>
<td>4-Bit Cascadable Full Adder with Carry-In, Carry-Out, and Overflow</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADD8</td>
<td>8-Bit Cascadable Full Adder with Carry-In, Carry-Out, and Overflow</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADD16</td>
<td>16-Bit Cascadable Full Adder with Carry-In, Carry-Out, and Overflow</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADSU1</td>
<td>1-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADSU4</td>
<td>4-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out, and Overflow</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Micro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>ADSU8</td>
<td>8-Bit Cascadable Adder/Subtracter with Carry-In, Carry-Out, and Overflow</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/ XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------</td>
<td>-----------------</td>
<td>------------</td>
<td>-----------</td>
<td>------------------------</td>
<td>----------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>ADSU16</td>
<td>16-Bit Cascadable Adder/Subtractor with Carry-In, Carry-Out, and Overflow</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>MULT18X18</td>
<td>18 x 18 Signed Multiplier</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MULT18X18S</td>
<td>18 x 18 Signed Multiplier – Registered Version</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Buffers

The buffers in this section route high fanout signals, 3-state signals, and clocks inside a PLD device. The “Input/Output Functions” section covers off-chip interfaces.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUF</td>
<td>General Purpose Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>BUF4</td>
<td>4-Bit General Purpose Buffer</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>BUF8</td>
<td>8-Bit General Purpose Buffer</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>BUF16</td>
<td>16-Bit General Purpose Buffer</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>BUFCF</td>
<td>Fast Connect Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFE</td>
<td>Internal 3-State Buffer with Active High Enable</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFE4</td>
<td>Internal 3-State Buffer with Active High Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFE8</td>
<td>Internal 3-State Buffer with Active High Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFE16</td>
<td>Internal 3-State Buffer with Active High Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFG</td>
<td>Global Clock Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUPGCE</td>
<td>Global Clock MUX with Clock Enable and Output State 0</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BUPGCE_1</td>
<td>Global Clock MUX Buffer with Clock Enable and Output State 1</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BUFGDLL</td>
<td>Clock Delay Locked Loop Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFGMUX</td>
<td>Global Clock MUX Buffer with Output State 0</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BUFGMUX_1</td>
<td>Global Clock MUX with Output State 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BUFGP</td>
<td>Primary Global Buffer for Driving Clocks or Longlines (Four per PLD Device)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFGSR</td>
<td>Global Set/Reset Input Buffer</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>BUFGTS</td>
<td>Global 3-State Input Buffer</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>BUFT</td>
<td>Internal 3-State Buffer with Active-Low Enable</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFT4</td>
<td>Internal 3-State Buffer with Active-Low Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFT8</td>
<td>Internal 3-State Buffer with Active-Low Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BUFT16</td>
<td>Internal 3-State Buffer with Active-Low Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

a. Not supported for XC9500XL and XC9500XV devices.
b. Not supported for XC9500XL and XC9500XV devices.
c. Not supported for XC9500XL and XC9500XV devices.
d. Not supported for XC9500XL and XC9500XV devices.
e. Not supported for XC9500XL and XC9500XV devices.
f. Not supported for XC9500XL and XC9500XV devices.
g. Not supported for XC9500XL and XC9500XV devices.

h. Not supported for XC9500XL and XC9500XV devices.
Comparators

Following is a list of comparators.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP2</td>
<td>2-Bit Identity Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMP4</td>
<td>4-Bit Identity Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMP8</td>
<td>8-Bit Identity Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMP16</td>
<td>16-Bit Identity Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMPM2</td>
<td>2-Bit Magnitude Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMPM4</td>
<td>4-Bit Magnitude Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMPM8</td>
<td>8-Bit Magnitude Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMPM16</td>
<td>16-Bit Magnitude Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>COMPMC8</td>
<td>8-Bit Magnitude Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COMPMC16</td>
<td>16-Bit Magnitude Comparator</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Counters

There are six types of counters with various synchronous and asynchronous inputs. The name of the counter defines the modulo or bit size, the counter type, and which control functions are included. The counter naming convention is shown in the following figure.

Counter Naming Convention

A carry-lookahead design accommodates large counters without extra gating. On TTL 7400-type counters with trickle clock enable (ENT), parallel clock enable (ENP), and ripple carry-out (RCO), both the ENT and ENP inputs must be High to count. ENT is propagated forward to enable RCO, which produces a High output with the approximate duration of the QA output. The following figure illustrates a carry-lookahead design.

Carry-Lookahead Design
The RCO output of the first stage of the ripple carry is connected to the ENP input of the second stage and all subsequent stages. The RCO output of the second stage and all subsequent stages is connected to the ENT input of the next stage. The ENT of the second stage is always enabled/tied to VCC. CE is always connected to the ENT input of the first stage. This cascading method allows the first stage of the ripple carry to be built as a prescaler. In other words, the first stage is built to count very fast.

**Note:** For counters, do not use TC (or any other gated signal) as a clock. Possible glitches may not always allow for a proper setup time when using gated signals.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB2CE</td>
<td>2-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB4CE</td>
<td>4-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB8CE</td>
<td>8-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB16CE</td>
<td>16-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB2CLE</td>
<td>2-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB4CLE</td>
<td>4-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB8CLE</td>
<td>8-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB16CLE</td>
<td>16-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB2CLED</td>
<td>2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB4CLED</td>
<td>2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB8CLED</td>
<td>2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB16CLED</td>
<td>2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB2RE</td>
<td>2-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB4RE</td>
<td>4-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB8RE</td>
<td>8-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CB16RE</td>
<td>16-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>----------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>CB2RLE</td>
<td>2-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB4RLE</td>
<td>4-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB8RLE</td>
<td>8-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB16RLE</td>
<td>16-Bit Loadable Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB2X1</td>
<td>2-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB4X1</td>
<td>4-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB8X1</td>
<td>8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB16X1</td>
<td>16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB2X2</td>
<td>2-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB4X2</td>
<td>4-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB8X2</td>
<td>8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CB16X2</td>
<td>16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2CE</td>
<td>2-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD4CE</td>
<td>4-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8CE</td>
<td>8-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16CE</td>
<td>16-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2CLE</td>
<td>2-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>CBD4CLE</td>
<td>4-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8CLE</td>
<td>8-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16CLE</td>
<td>16-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2CLED</td>
<td>2-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD4CED</td>
<td>4-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8CLED</td>
<td>8-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16CLED</td>
<td>16-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2RE</td>
<td>2-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD4RE</td>
<td>4-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8RE</td>
<td>8-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16RE</td>
<td>16-Bit Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2RLE</td>
<td>2-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD4RLE</td>
<td>4-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8RLE</td>
<td>8-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16RLE</td>
<td>16-Bit Loadable Cascadable Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2X1</td>
<td>2-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/ XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>---------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>CBD4X1</td>
<td>4-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8X1</td>
<td>8-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16X1</td>
<td>16-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD2X2</td>
<td>2-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD4X2</td>
<td>4-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD8X2</td>
<td>8-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CBD16X2</td>
<td>16-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CC8CE</td>
<td>8-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC16CE</td>
<td>16-Bit Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC8CLE</td>
<td>8-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC16CLE</td>
<td>16-Bit Loadable Cascadable Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC8CLED</td>
<td>8-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC16CLED</td>
<td>16-Bit Loadable Cascadable Bidirectional Binary Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC8RE</td>
<td>8-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CC16RE</td>
<td>16-Bit Cascadable Binary Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CD4CE</td>
<td>4-Bit Cascadable BCD Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CD4CLE</td>
<td>4-Bit Loadable Cascadable BCD Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CD2X2CE</td>
<td>2-Bit Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>CD4RE</td>
<td>4-Bit Cascadable BCD Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CD4RLE</td>
<td>4-Bit Cascadable BCD Counter with Clock Enable And Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CDD4CLE</td>
<td>4-Bit Loadable Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CD4RE</td>
<td>4-Bit Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CD4RLE</td>
<td>4-Bit Loadable Cascadable BCD Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CD4RE</td>
<td>4-Bit Cascadable BCD Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CJ4E</td>
<td>4-Bit Johnson Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CJ5CE</td>
<td>5-Bit Johnson Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CJ8CE</td>
<td>8-Bit Johnson Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CJD4CE</td>
<td>4-Bit Dual Edge Triggered Johnson Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CJD5CE</td>
<td>5-Bit Dual Edge Triggered Johnson Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CJD8CE</td>
<td>8-Bit Dual Edge Triggered Johnson Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CJD4RE</td>
<td>4-Bit Dual Edge Triggered Johnson Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CJD5RE</td>
<td>5-Bit Dual Edge Triggered Johnson Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CJD8RE</td>
<td>8-Bit Dual Edge Triggered Johnson Counter with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CJ4RE</td>
<td>4-Bit Johnson Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CJ5RE</td>
<td>5-Bit Johnson Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CJ8RE</td>
<td>8-Bit Johnson Counter with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CR8CE</td>
<td>8-Bit Negative-Edge Binary Ripple Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CR16CE</td>
<td>16-Bit Negative-Edge Binary Ripple Counter with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>CRD8CE</td>
<td>8-Bit Dual-Edge Triggered Binary Ripple Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>CRD16CE</td>
<td>16-Bit Dual-Edge Triggered Binary Ripple Counter with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
</tbody>
</table>
Decoders

Decoder names, shown in the following figure, indicate the number of inputs and outputs and whether or not an enable is available. Decoders with an enable can be used as multiplexers.

![Decoder Diagram]

### Decoder Naming Convention

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, II E</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2_4E</td>
<td>2- to 4-Line Decoder/Demultiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>D3_8E</td>
<td>3- to 8-Line Decoder/Demultiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>D4_16E</td>
<td>4- to 16-Line Decoder/Demultiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>DEC_CC4</td>
<td>4-Bit Active Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DEC_CC8</td>
<td>8-Bit Active Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DEC_CC16</td>
<td>16-Bit Active Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### Edge Decoders

Edge decoders are open-drain wired AND gates that are available in different bit sizes.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECODE4</td>
<td>4-Bit Active-Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECODE8</td>
<td>8-Bit Active-Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECODE16</td>
<td>16-Bit Active-Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECODE32</td>
<td>32-Bit Active-Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECODE64</td>
<td>64-Bit Active-Low Decoder</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Flip-Flops

There are three types of flip-flops (D, J-K, toggle) with various synchronous and asynchronous inputs. Some are available with inverted clock inputs and/or the ability to set in response to global set/reset rather than reset. The naming convention shown in the following figure provides a description for each flip-flop. D-type flip-flops are available in multiples of up to 16 in one macro.

![Flip-Flop Diagram](image)

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD</td>
<td>D Flip-Flop</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD_1</td>
<td>D Flip-Flop with Negative-Edge Clock</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD4</td>
<td>Multiple D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD8</td>
<td>Multiple D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD16</td>
<td>Multiple D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD4CE</td>
<td>4-Bit Data Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD8CE</td>
<td>8-Bit Data Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD16CE</td>
<td>16-Bit Data Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD4RE</td>
<td>4-Bit Data Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD8RE</td>
<td>8-Bit Data Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FD16RE</td>
<td>16-Bit Data Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDC</td>
<td>D Flip-Flop with Asynchronous Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDC_1</td>
<td>D Flip-Flop with Negative-Edge Clock and Asynchronous Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDCE</td>
<td>D Flip-Flop with Clock Enable and Asynchronous Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDCE_1</td>
<td>D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDCP</td>
<td>D Flip-Flop with Asynchronous Preset and Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDCP_1</td>
<td>D Flip-Flop with Negative-Edge Clock and Asynchronous Preset and Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>FDCPE</td>
<td>D Flip-Flop with Clock Enable and Asynchronous Preset and Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDCPE_3</td>
<td>D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Preset and Clear</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDD</td>
<td>Dual Edge Triggered D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD4</td>
<td>Multiple Dual Edge Triggered D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD8</td>
<td>Multiple Dual Edge Triggered D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD16</td>
<td>Multiple Dual Edge Triggered D Flip-Flop</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD4CE</td>
<td>4-Bit Dual Edge Triggered Data Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD8CE</td>
<td>8-Bit Dual Edge Triggered Data Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD16CE</td>
<td>16-Bit Dual Edge Triggered Data Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD4RE</td>
<td>4-Bit Dual Edge Triggered Data Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD8RE</td>
<td>8-Bit Dual Edge Triggered Data Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDD16RE</td>
<td>16-Bit Dual Edge Triggered Data Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDC</td>
<td>D Dual Edge Triggered Flip-Flop with Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDCE</td>
<td>Dual Edge Triggered D Flip-Flop with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDDCP</td>
<td>Dual Edge Triggered D Flip-Flop Asynchronous Preset and Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDDCPE</td>
<td>Dual Edge Triggered D Flip-Flop with Asynchronous Preset and Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDDP</td>
<td>Dual Edge Triggered D Flip-Flop with Asynchronous Preset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDPE</td>
<td>Dual Edge Triggered D Flip-Flop with Clock Enable and Asynchronous Preset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDDR</td>
<td>Dual Edge Triggered D Flip-Flop with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDRCPE</td>
<td>Dual Data Rate D Flip-Flop with Clock Enable and Asynchronous Preset and Clear</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDDRE</td>
<td>Dual Edge Triggered D Flip-Flop with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDRRSE</td>
<td>Dual Data Rate D Flip-Flop with Clock Enable and Synchronous Reset and Set</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDDRS</td>
<td>Dual Edge Triggered D Flip-Flop with Synchronous Reset and Set</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>FDDRSE</td>
<td>Dual Edge Triggered D Flip-Flop with Synchronous Reset and Set and Clock Enable</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDS</td>
<td>Dual Edge Triggered D Flip-Flop with Synchronous Set</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDSE</td>
<td>D Flip-Flop with Clock Enable and Synchronous Set</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDDSRE</td>
<td>Dual Edge Triggered D Flip-Flop with Synchronous Set and Reset and Clock Enable</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FDE</td>
<td>D Flip-Flop with Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDE_1</td>
<td>D Flip-Flop with Negative-Edge Clock and Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDP</td>
<td>D Flip-Flop with Asynchronous Preset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDP_1</td>
<td>D Flip-Flop with Negative-Edge Clock and Asynchronous Preset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDPE</td>
<td>D Flip-Flop with Clock Enable and Asynchronous Preset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>FDPE_1</td>
<td>D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Preset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDR</td>
<td>D Flip-Flop with Synchronous Reset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDR_1</td>
<td>D Flip-Flop with Negative-Edge Clock and Synchronous Reset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDRE</td>
<td>D Flip-Flop with Clock Enable and Synchronous Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDRE_1</td>
<td>D Flip-Flop with Negative-Edge Clock, Clock Enable, and Synchronous Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDRS</td>
<td>D Flip-Flop with Synchronous Reset and Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDRS_1</td>
<td>D Flip-Flop with Negative-Edge Clock, Synchronous Reset and Set, and Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDRSE</td>
<td>D Flip-Flop with Synchronous Reset and Set and Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDRSE_1</td>
<td>D Flip-Flop with Synchronous Reset, Set and Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDS</td>
<td>D Flip-Flop with Synchronous Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDS_1</td>
<td>D Flip-Flop with Negative-Edge Clock and Synchronous Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDSE</td>
<td>D Flip-Flop with Clock Enable and Synchronous Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDSE_1</td>
<td>D Flip-Flop with Negative-Edge Clock, Clock Enable, and Synchronous Set</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FDSR</td>
<td>D Flip-Flop with Synchronous Set and Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FDSRE</td>
<td>D Flip-Flop with Synchronous Set and Reset and Clock Enable</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKC</td>
<td>J-K Flip-Flop with Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>FJKCE</td>
<td>J-K Flip-Flop with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKCP</td>
<td>J-K Flip-Flop with Asynchronous Clear and Preset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKCPE</td>
<td>J-K Flip-Flop with Asynchronous Clear and Preset and Clock Enable</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKP</td>
<td>J-K Flip-Flop with Asynchronous Preset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKPE</td>
<td>J-K Flip-Flop with Asynchronous Preset and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKRSE</td>
<td>J-K Flip-Flop with Clock Enable and Asynchronous Reset and Set</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FJKSRE</td>
<td>J-K Flip-Flop with Clock Enable and Asynchronous Set and Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTC</td>
<td>Toggle Flip-Flop with Toggle Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTCE</td>
<td>Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTCLE</td>
<td>Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTCLEX</td>
<td>Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTCP</td>
<td>Toggle Flip-Flop with Toggle Enable and Asynchronous Clear and Preset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>FTCPE</td>
<td>Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTCPLE</td>
<td>Loadable Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear and Preset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTDCE</td>
<td>Dual Edge Triggered Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FTDCELE</td>
<td>Dual Edge Triggered Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FTDCELEX</td>
<td>Dual Edge Triggered Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FTDCEP</td>
<td>Toggle Flip-Flop with Toggle Enable and Asynchronous Clear and Preset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>FTDRE</td>
<td>Dual Edge Triggered Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>FTDRESLE</td>
<td>Dual Edge Triggered Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTP</td>
<td>Toggle Flip-Flop with Toggle Enable and Asynchronous Preset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTPPE</td>
<td>Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Preset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>FTPLE</td>
<td>Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Preset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTRSE</td>
<td>Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTRSLE</td>
<td>Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTSRE</td>
<td>Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>FTSRLE</td>
<td>Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
</tbody>
</table>
## General

General elements include FPGA configuration functions, oscillators, boundary scan logic, and other functions not classified in other sections.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSCAN_SPARTAN2</td>
<td>Spartan-II Boundary Scan Logic Control Circuit</td>
<td>Primitive(^a)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BSCAN_SPARTAN3</td>
<td>Spartan-3 Boundary Scan Logic Control Circuit</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BSCAN_VIRTEX</td>
<td>Virtex Boundary Scan Logic Control Circuit</td>
<td>Primitive(^b)</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BSCAN_VIRTEX2</td>
<td>Virtex2 Boundary Scan Logic Control Circuit</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CAPTURE_SPARTAN_N2</td>
<td>Spartan-II Register State Capture for Bitstream Readback</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CAPTURE_SPARTAN_N3</td>
<td>Spartan-3 Register State Capture for Bitstream Readback</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CAPTURE_VIRTEX</td>
<td>Virtex Register State Capture for Bitstream Readback</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CAPTURE_VIRTEX2</td>
<td>Virtex-II Register State Capture for Bitstream Readback</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV2</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV4</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV6</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV8</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV10</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV12</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV14</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV16</td>
<td>Global Clock Divider</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV2R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV4R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV6R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV8R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV10R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV12R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV14R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV16R</td>
<td>Global Clock Divider with Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV2RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLK_DIV4RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>---------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>CLK_DIV6RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV8RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV10RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV12RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV14RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV16RSD</td>
<td>Global Clock Divider with Synchronous Reset and Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV2SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV4SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV6SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV8SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV10SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV12SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV14SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLK_DIV16SD</td>
<td>Global Clock Divider with Start Delay</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
</tr>
<tr>
<td>CLKDLL</td>
<td>Clock Delay Locked Loop</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLKDLLLE</td>
<td>Clock Delay Locked Loop with Expanded Output</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CLKDLLHF</td>
<td>High Frequency Clock Delay Locked Loop</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DCM</td>
<td>Digital Clock Manager</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GND</td>
<td>Ground-Connection Signal Tag</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>ICAP_VIRTEX2</td>
<td>User Interface to Virtex-II Internal Configuration Access Port</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>JTAGPPC</td>
<td>JTAG Primitive for the Power PC</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>KEEPER</td>
<td>KEEPER Symbol</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT1</td>
<td>1-Bit Look-Up-Table with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT2</td>
<td>2-Bit Look-Up-Table with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT3</td>
<td>3-Bit Look-Up-Table with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT4</td>
<td>4-Bit Look-Up-Table with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT1_D</td>
<td>1-Bit Look-Up-Table with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/ XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>LUT2_D</td>
<td>2-Bit Look-Up-Table with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT3_D</td>
<td>3-Bit Look-Up-Table with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT4_D</td>
<td>4-Bit Look-Up-Table with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT1_L</td>
<td>1-Bit Look-Up-Table with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT2_L</td>
<td>2-Bit Look-Up-Table with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT3_L</td>
<td>3-Bit Look-Up-Table with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LUT4_L</td>
<td>4-Bit Look-Up-Table with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PPC405</td>
<td>Primitive for the Power PC Core.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PULLDOWN</td>
<td>Resistor to GND for Input Pads</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PULLUP</td>
<td>Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROC</td>
<td>Reset On Configuration</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARTBUFF_architecture</td>
<td>VHDL Simulation of FPGA Designs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARTUP_SPARTA_N2</td>
<td>Spartan-II User Interface to Global Clock, Reset, and 3-State Controls.</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARTUP_SPARTA_N3</td>
<td>Spartan-3 User Interface to Global Clock, Reset, and 3-State Controls</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARTUP_VIRTEX</td>
<td>Virtex User Interface to Global Clock, Reset, and 3-State Controls.</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARTUP_VIRTEX2</td>
<td>Virtex-II User Interface to Global Clock, Reset, and 3-State Controls.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TOC</td>
<td>Three-State On Configuration</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TOCBUF</td>
<td>Three-State On Configuration Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VCC</td>
<td>VCC-Connection Signal Tag</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

a. Primitive is supported for Spartan-II, but not for Spartan-IIE, which is supported by BSCAN_VIRTEX.
b. Primitive is supported for Spartan-IIE, but not for Spartan-II, which is supported by BSCAN_SPARTAN2.
c. Supported for Spartan-IIE and Virtex-E devices only.
d. Supported for Spartan-IIE and Virtex-E devices only.
e. For Virtex E, use CLKDLLHF in HF mode. In LF mode, both the separate CLKDLLE and CLKDLL primitive can be used.
f. Not supported for Virtex-II. Supported for Virtex-II PRO only.
g. Under XC9500, the Primitive implementation applies only to XC9500XL/XV.
h. The Primitive in the Virtex-II field is supported for Virtex-II PRO only.
i. The Primitive in the field marked Spartan IIE is supported only for Spartan-II but not for Spartan-IIE, the latter of which is supported by STARTUP_VIRTEX.
j. The Primitive in the Spartan IIE field is supported for Spartan-IIE, but not for Spartan-II, which is supported by STARTUP_SPARTAN2.
## Input Latches

Single and multiple input latches can hold transient data entering a chip. Input latches use the same naming convention as I/O flip-flops.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILD</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILD4</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILD8</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILD16</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILD.1</td>
<td>Transparent Input DataLatch with Inverted Gate</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDI</td>
<td>Transparent Input DataLatch (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDI.1</td>
<td>Transparent Input DataLatch (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDX</td>
<td>Transparent Input DataLatch with Inverted Gate</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDX4</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDX8</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDX16</td>
<td>Transparent Input DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDX.3</td>
<td>Transparent Input DataLatch with Inverted Gate</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDXI</td>
<td>Transparent Input DataLatch (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ILDXI.1</td>
<td>Transparent Input DataLatch (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Input/Output Flip-Flops

Input/Output flip-flops are configured in IOBs. They include flip-flops whose outputs are enabled by 3-state buffers, flip-flops that can be set upon global set/reset rather than reset, and flip-flops with inverted clock inputs. The naming convention specifies each flip-flop function and is illustrated in the following figure.

![Input/Output Flip-Flop Naming Convention](image)

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFD</td>
<td>Single- and Multiple-Input D Flip-Flop</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFD_1</td>
<td>Input D Flip-Flop with Inverted Clock</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFD4</td>
<td>Single- and Multiple-Input D Flip-Flop</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFD8</td>
<td>Single- and Multiple-Input D Flip-Flop</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFD16</td>
<td>Single- and Multiple-Input D Flip-Flop</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDDRCPE</td>
<td>Dual Data Rate Input D Flip-Flop with Clock Enable and Asynchronous Preset and Clear</td>
<td>No Primitive</td>
<td>No Primitive</td>
<td>No Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IFDDRSE</td>
<td>Dual Data Rate Input D Flip-Flop with Synchronous Reset and Set and Clock Enable</td>
<td>No Primitive</td>
<td>No Primitive</td>
<td>No Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IFDI</td>
<td>Input D Flip-Flop (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDI_1</td>
<td>Input D Flip-Flop with Inverted Clock (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDX</td>
<td>Single- and Multiple-Input D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDX_1</td>
<td>Input D Flip-Flop with Inverted Clock and Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDX4</td>
<td>Single- and Multiple-Input D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDX8</td>
<td>Single- and Multiple-Input D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IFDX16</td>
<td>Single- and Multiple-Input D Flip-Flops with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>IFDXI</td>
<td>Input D Flip-Flop with Clock Enable (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFD</td>
<td>Single- and Multiple-Output D Flip-Flops</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFD4</td>
<td>Single- and Multiple-Output D Flip-Flops</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFD8</td>
<td>Single- and Multiple-Output D Flip-Flops</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFD16</td>
<td>Single- and Multiple-Output D Flip-Flops</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFD,1</td>
<td>Output D Flip-Flop with Inverted Clock</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDDRCPE</td>
<td>Dual Data Rate Output D Flip-Flop with Clock Enable and Asynchronous Preset and Clear</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDDRRSE</td>
<td>Dual Data Rate Output D Flip-Flop with Synchronous Reset and Set and Clock Enable</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDDRTCPE</td>
<td>Dual Data Rate D Flip-Flop with Active-Low 3-State Output Buffer, Clock Enable, and Asynchronous Preset and Clear</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDDRTRSE</td>
<td>Dual Data Rate D Flip-Flop with Active-Low 3-State Output Buffer, Synchronous Reset and Set, and Clock Enable</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDE</td>
<td>D Flip-Flop with Active-High Enable Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDE4</td>
<td>D Flip-Flop with Active-High Enable Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDE8</td>
<td>D Flip-Flop with Active-High Enable Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDE16</td>
<td>D Flip-Flop with Active-High Enable Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDE,1</td>
<td>D Flip-Flop with Active-High Enable Output Buffer and Inverted Clock</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDI</td>
<td>Output D Flip-Flop (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFD,1</td>
<td>Output D Flip-Flop with Inverted Clock (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDT</td>
<td>Single and Multiple D Flip-Flop with Active-Low 3-State Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDT4</td>
<td>Single and Multiple D Flip-Flop with Active-Low 3-State Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDT8</td>
<td>Single and Multiple D Flip-Flop with Active-Low 3-State Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDT16</td>
<td>Single and Multiple D Flip-Flop with Active-Low 3-State Output Buffers</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFDT,1</td>
<td>D Flip-Flop with Active-Low 3-State Output Buffer and Inverted Clock</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/ XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>----------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>OFDX</td>
<td>Single- and Multiple- Output D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OFDX4</td>
<td>Single- and Multiple- Output D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OFDX8</td>
<td>Single- and Multiple- Output D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OFDX16</td>
<td>Single- and Multiple- Output D Flip-Flop with Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OFDX_1</td>
<td>Output D Flip-Flop with Inverted Clock and Clock Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OFDXI</td>
<td>Output D Flip-Flop with Clock Enable (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OFDXI_1</td>
<td>Output D Flip-Flop with Inverted Clock and Clock Enable (Asynchronous Preset)</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
**Input/Output Functions**

Input/Output Block (IOB) resources are configured into various I/O primitives and macros for convenience, such as output buffers (s) and output buffers with an enable (OBUFEs). Pads used to connect the circuit to PLD device pins are also included.

Virtex, Virtex-E, Spartan-II, Spartan-IIE, Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X have multiple variants (Primitives) to choose from for each SelectIO buffer. The I/O interface for each variant corresponds to a specific I/O standard.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT_AURORA_n</td>
<td>Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT_CUSTOM</td>
<td>Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT ETHERNET_n</td>
<td>Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT_FIBRE_CHAN_n</td>
<td>Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT_INFINITYBAND_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT_XAUI_n</td>
<td>Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_10GE_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_10GFC_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_AURORA_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_AURORAX_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_CUS-TOM</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_OC48_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_OCI92_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_PCI_EXPRESS_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GT10_XAUI_n</td>
<td>10-Gigabit Transceiver for High-Speed I/O.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUF</td>
<td>Single- and Multiple-Input Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>IBUF4</td>
<td>Single- and Multiple-Input Buffer</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>IBUF8</td>
<td>Single- and Multiple-Input Buffer</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>IBUF16</td>
<td>Single- and Multiple-Input Buffer</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>IBUF selectIO</td>
<td>Single Input Buffer with Selectable I/O Interface (multiple primitives)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFS5</td>
<td>Differential Signaling Input Buffer with Selectable I/O Interface</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG</td>
<td>Dedicated Input Buffer with Selectable I/O Interface</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>IBUF_DS</td>
<td>Dedicated Input Buffer with Selectable I/O Interface (multiple primitives)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFDS</td>
<td>Dedicated Differential Signaling Input Buffer with Selectable I/O Interface</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFDS_DIFF_OUT</td>
<td>Differential I/O Input Buffer with Differential Outputs</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF</td>
<td>Bi-Directional Buffer with Selectable I/O Interface (multiple primitives)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_SELECTIO</td>
<td>Bi-Directional Buffer with Selectable I/O Interface (multiple primitives)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IOPAD</td>
<td>Single- and Multiple-Input/Output Pad</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>IOPAD4</td>
<td>Single- and Multiple-Input/Output Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>IOPAD8</td>
<td>Single- and Multiple-Input/Output Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>IOPAD16</td>
<td>Single- and Multiple-Input/Output Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>IPAD</td>
<td>Single- and Multiple-Input Pad</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>IPAD4</td>
<td>Single- and Multiple-Input Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>IPAD8</td>
<td>Single- and Multiple-Input Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>IPAD16</td>
<td>Single- and Multiple-Input Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUF_SELECTIO</td>
<td>Single 3-State Output Buffer with Active-Low Output Enable and Selectable I/O Interface (multiple primitives)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OBUF</td>
<td>Single- and Multiple-Output Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OBUF4</td>
<td>Single- and Multiple-Output Buffer</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUF8</td>
<td>Single- and Multiple-Output Buffer</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUF16</td>
<td>Single- and Multiple-Output Buffer</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUFDS</td>
<td>Differential Signaling Output Buffer with Selectable I/O Interface</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OBUF_E</td>
<td>3-State Output Buffers with Active-High Output Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUF_E4</td>
<td>3-State Output Buffers with Active-High Output Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------------------</td>
<td>---------------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>OBUFES</td>
<td>3-State Output Buffers with Active-High Output Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUFES16</td>
<td>3-State Output Buffers with Active-High Output Enable</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUFST</td>
<td>Single and Multiple 3-State Output Buffer with Active Low Output Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OBUFST_SELIO</td>
<td>Single 3-State Output Buffer with Active-Low Output Enable and Selectable I/O Interface (multiple primitives)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OBUFST4</td>
<td>Single and Multiple 3-State Output Buffer with Active Low Output Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUFST8</td>
<td>Single and Multiple 3-State Output Buffer with Active Low Output Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUFST16</td>
<td>Single and Multiple 3-State Output Buffer with Active Low Output Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OBUFSTDS</td>
<td>3-State Output Buffer with Differential Signaling, Active-Low Output Enable, and Selectable I/O Interface</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OPAD</td>
<td>Single- and Multiple-Output Pad</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OPAD4</td>
<td>Multiple-Output Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OPAD8</td>
<td>Multiple-Output Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OPAD16</td>
<td>Multiple-Output Pad</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>UPAD</td>
<td>Connects the I/O Node of an IOB to the Internal PLD Circuit</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Latches

Latches (LD) are available for all architectures.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/ XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>Transparent DataLatch</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>LD_1</td>
<td>Transparent DataLatch with Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LD4</td>
<td>Multiple Transparent DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>LD8</td>
<td>Multiple Transparent DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>LD16</td>
<td>Multiple Transparent DataLatch</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>LD4CE</td>
<td>Transparent DataLatch with Asynchronous Clear and Gate Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LD8CE</td>
<td>Transparent DataLatch with Asynchronous Clear and Gate Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LD16CE</td>
<td>Transparent DataLatch with Asynchronous Clear and Gate Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDC</td>
<td>Transparent DataLatch with Asynchronous Clear and Gate Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>LDC_1</td>
<td>Transparent DataLatch with Asynchronous Clear and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDCE</td>
<td>Transparent DataLatch with Asynchronous Clear and Gate Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDCE_1</td>
<td>Transparent DataLatch with Asynchronous Clear, Gate Enable, and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDCP</td>
<td>Transparent DataLatch with Asynchronous Clear and Preset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>LDCP_1</td>
<td>Transparent DataLatch with Asynchronous Clear and Preset and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDCPE</td>
<td>Transparent DataLatch with Asynchronous Clear and Preset and Gate Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDCPE_1</td>
<td>Transparent DataLatch with Asynchronous Clear and Preset, Gate Enable, and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDE</td>
<td>Transparent DataLatch with Gate Enable, and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDE_1</td>
<td>Transparent DataLatch with Gate Enable and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDG</td>
<td>Transparent DatagateLatch</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>LDG4</td>
<td>Multiple Transparent DatagateLatch</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDG8</td>
<td>Multiple Transparent DatagateLatch</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDG16</td>
<td>Multiple Transparent DatagateLatch</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>LDP</td>
<td>Transparent Data Latch with Asynchronous Preset</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>LDP_1</td>
<td>Transparent Data Latch with Asynchronous Preset and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDPE</td>
<td>Transparent Data Latch with Asynchronous Preset and Gate Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LDPE_1</td>
<td>Transparent Data Latch with Asynchronous Preset, Gate Enable, and Inverted Gate</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Logic Primitives

Combinatorial logic gates that implement the basic Boolean functions are available in all architectures with up to five inputs in all combinations of inverted and non-inverted inputs, and with six to nine inputs non-inverted.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND2</td>
<td>2-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>AND2B1</td>
<td>2-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND2B2</td>
<td>2-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND3</td>
<td>3-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>AND3B1</td>
<td>3-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND3B2</td>
<td>3-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND3B3</td>
<td>3-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND4</td>
<td>4-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>AND4B1</td>
<td>4-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND4B2</td>
<td>4-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND4B3</td>
<td>4-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND4B4</td>
<td>4-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND5</td>
<td>5-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>AND5B1</td>
<td>5-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND5B2</td>
<td>5-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND5B3</td>
<td>5-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND5B4</td>
<td>5-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>AND5B5</td>
<td>5-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>AND6</td>
<td>6-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>AND7</td>
<td>7-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>AND8</td>
<td>8-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>AND9</td>
<td>9-Input AND Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>AND10</td>
<td>10-Input AND Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>AND12</td>
<td>12-Input AND Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>AND16</td>
<td>16-Input AND Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INV</td>
<td>Single and Multiple Inverters</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>INV4</td>
<td>Single and Multiple Inverters</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>INV8</td>
<td>Single and Multiple Inverters</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>INV16</td>
<td>Single and Multiple Inverters</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>MULT_AND</td>
<td>Fast Multiplier AND</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NAND2</td>
<td>2-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>NAND2B1</td>
<td>2-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND2B2</td>
<td>2-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND3</td>
<td>3-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>NAND3B1</td>
<td>3-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND3B2</td>
<td>3-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND3B3</td>
<td>3-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND4</td>
<td>4-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>NAND4B1</td>
<td>4-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND4B2</td>
<td>4-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND4B3</td>
<td>4-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND4B4</td>
<td>4-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>NAND5</td>
<td>5-Input NAND Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>NAND5B1</td>
<td>5-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND5B2</td>
<td>5-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND5B3</td>
<td>5-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND5B4</td>
<td>5-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND5B5</td>
<td>5-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND6</td>
<td>6-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND7</td>
<td>7-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND8</td>
<td>8-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND9</td>
<td>9-Input NAND Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NAND12</td>
<td>12-Input NAND Gate with Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NAND16</td>
<td>16-Input NAND Gate with Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOR2</td>
<td>2-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>NOR3</td>
<td>3-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>NOR4</td>
<td>4-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>NOR5</td>
<td>5-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>NOR6</td>
<td>6-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR7</td>
<td>7-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR8</td>
<td>8-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR9</td>
<td>9-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR12</td>
<td>12-Input NOR Gate with Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOR16</td>
<td>16-Input NOR Gate with Non-Inverted Inputs.</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOR2B1</td>
<td>2-Input NOR Gate with Inverted and Non-Inverted Inputs.</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>NOR2B2</td>
<td>2-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR3B1</td>
<td>3-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR3B2</td>
<td>3-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR3B3</td>
<td>3-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR4B1</td>
<td>4-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR4B2</td>
<td>4-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR4B3</td>
<td>4-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR4B4</td>
<td>4-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR5B1</td>
<td>5-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR5B2</td>
<td>5-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR5B3</td>
<td>5-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR5B4</td>
<td>5-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>NOR5B5</td>
<td>5-Input NOR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR2</td>
<td>2-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR2B1</td>
<td>2-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR2B2</td>
<td>2-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR3</td>
<td>3-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR3B1</td>
<td>3-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR3B2</td>
<td>3-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR3B3</td>
<td>3-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR4</td>
<td>4-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------</td>
<td>-----------------</td>
<td>----------</td>
<td>-----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>OR4B1</td>
<td>4-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR4B2</td>
<td>4-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR4B3</td>
<td>4-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>OR5B1</td>
<td>5-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR5B2</td>
<td>12-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR5B3</td>
<td>5-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR5B4</td>
<td>5-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR5B5</td>
<td>5-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR6</td>
<td>6-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR7</td>
<td>6-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR8</td>
<td>8-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR9</td>
<td>9-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>OR12</td>
<td>12-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OR16</td>
<td>16-Input OR Gate with Inverted and Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ORCY</td>
<td>OR with Carry Logic</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP3</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP3B1A</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP3B1B</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP3B2A</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP3B2B</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP4</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP4B3</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP4B4</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP4B1</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP4B2A</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SOP4B2B</td>
<td>Sum of Products</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>XNOR2</td>
<td>2-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>XNOR3</td>
<td>3-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>XNOR4</td>
<td>4-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>XNOR5</td>
<td>5-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XNOR6</td>
<td>6-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XNOR7</td>
<td>7-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XNOR8</td>
<td>8-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XNOR9</td>
<td>9-Input XNOR Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XOR2</td>
<td>2-Input XOR Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>XOR3</td>
<td>3-Input XOR Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>XOR4</td>
<td>4-Input XOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>XOR5</td>
<td>5-Input XOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
<tr>
<td>XOR6</td>
<td>6-Input XOR Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XOR7</td>
<td>7-Input XOR Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XOR8</td>
<td>8-Input XOR Gate with Non-Inverted Inputs</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XOR9</td>
<td>9-Input XOR Gate with Non-Inverted Inputs</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XORCY</td>
<td>XOR for Carry Logic with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>XORCY_D</td>
<td>XOR for Carry Logic with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>XORCY_L</td>
<td>XOR for Carry Logic with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Map Elements

Map elements are used in conjunction with logic symbols to constrain the logic to particular CLBs or particular F function generators.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMAP</td>
<td>F Function Generator Partitioning Control Symbol</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Memory Elements

In the Virtex, Virtex-E, Spartan-II, and Spartan-IIE, Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X architectures, a number of static RAMs are defined as primitives. These 16- or 32-word RAMs are 1, 2, 4, and 8 bits wide.

The Virtex, Virtex-E, Spartan-II, and Spartan-IIE, Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X architectures have dedicated blocks of on-chip 4096-bit single-port and dual-port synchronous RAM. Each port is configured to a specific data width. There are five single-port block RAM primitives and 30 dual-port block RAM primitives.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM16X1D</td>
<td>16-Deep by 1-Wide Static Dual Port Synchronous RAM</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X1D_1</td>
<td>16-Deep by 1-Wide Static Dual Port Synchronous RAM with Negative-Edge Clock</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X1S</td>
<td>16-Deep by 1-Wide Static Synchronous RAM</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X1S_1</td>
<td>16-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X2D</td>
<td>16-Deep by 2-Wide Static Dual Port Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X2S</td>
<td>16-Deep by 2-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM16X4D</td>
<td>16-Deep by 4-Wide Static Dual Port Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X4S</td>
<td>16-Deep by 4-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM16X8D</td>
<td>16-Deep by 8-Wide Static Dual Port Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM16X8S</td>
<td>16-Deep by 8-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>32-Deep by 1-Wide Static Dual Port Synchronous RAM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM32X1D_1</td>
<td>32-Deep by 1-Wide Static Dual Port Synchronous RAM with Negative-Edge Clock</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>32-Deep by 1-Wide Static Synchronous RAM</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM32X1S_1</td>
<td>32-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAM32X2D</td>
<td>32-Deep by 2-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>Primitive</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM32X2S</td>
<td>32-Deep by 2-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM32X4S</td>
<td>32-Deep by 4-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM32X8S</td>
<td>32-Deep by 8-Wide Static Synchronous RAM</td>
<td>Macro</td>
<td>No</td>
<td>Macro</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM64X1D</td>
<td>64-Deep by 1-Wide Dual Port Synchronous RAM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM64X1D_1</td>
<td>64-Deep by 1-Wide Dual Port Synchronous RAM with Negative-Edge Clock</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>64-Deep by 1-Wide Static Synchronous RAM</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XCV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>------------</td>
<td>------------</td>
<td>------------------------</td>
<td>---------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>RAM64X1S_1</td>
<td>64-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM64X2S</td>
<td>64-Deep by 2-Wide Static Synchronous RAM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM128X1S</td>
<td>128-Deep by 1-Wide Static Synchronous RAM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAM128X1S_1</td>
<td>128-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAMB4_Sm_Sn</td>
<td>4096-Bit Dual-Port Synchronous Block RAM with Port Width (m or n) Configured to 1, 2, 4, 8, or 16 Bits</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAMB4_Sn</td>
<td>4096-Bit Single-Port Synchronous Block RAM with Port Width (n) Configured to 1, 2, 4, 8, or 16 Bits</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAMB16_Sm_Sn</td>
<td>16384-Bit Data Memory and 2048-Bit Parity Memory, Dual-Port Synchronous Block RAM with Port Width (m or n) Configured to 1, 2, 4, 9, 18, or 36 Bits</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAMB16_Sn</td>
<td>16384-Bit Data Memory and 2048-Bit Parity Memory, Single-Port Synchronous Block RAM with Port Width (n) Configured to 1, 2, 4, 9, 18, or 36 Bits</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROC</td>
<td>Reset On Configuration</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROCBUF</td>
<td>Reset On Configuration Buffer</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROM16X1</td>
<td>16-Deep by 1-Wide ROM</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROM32X1</td>
<td>32-Deep by 1-Wide ROM</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROM64X1</td>
<td>64-Deep by 1-Wide ROM</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROM128X1</td>
<td>128-Deep by 1-Wide ROM</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROM256X1</td>
<td>256-Deep by 1-Wide ROM</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Multiplexers

The multiplexer naming convention shown in the following figure indicates the number of inputs and outputs and whether or not an enable is available.

**Multiplexer Naming Convention**

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2_1</td>
<td>2-to-1 Multiplexer</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>M2_1B1</td>
<td>2-to-1 Multiplexer with D0 Inverted</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>M2_1B2</td>
<td>2-to-1 Multiplexer with D0 and D1 Inverted</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>M2_1E</td>
<td>2-to-1 Multiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>M4_1E</td>
<td>4-to-1 Multiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>M8_1E</td>
<td>8-to-1 Multiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>M16_1E</td>
<td>16-to-1 Multiplexer with Enable</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>MUXCY</td>
<td>2-to-1 Multiplexer for Carry Logic with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXCY_D</td>
<td>2-to-1 Multiplexer for Carry Logic with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXCY_L</td>
<td>2-to-1 Multiplexer for Carry Logic with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF5</td>
<td>2-to-1 Lookup Table Multiplexer with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF5_D</td>
<td>2-to-1 Lookup Table Multiplexer with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF5_L</td>
<td>2-to-1 Lookup Table Multiplexer with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF6</td>
<td>2-to-1 Lookup Table Multiplexer with General Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF6_D</td>
<td>2-to-1 Lookup Table Multiplexer with Dual Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF6_L</td>
<td>2-to-1 Lookup Table Multiplexer with Local Output</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF7</td>
<td>2-to-1 Lookup Table Multiplexer with General Output</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF7_D</td>
<td>2-to-1 Lookup Table Multiplexer with Dual Output</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF7_L</td>
<td>2-to-1 Lookup Table Multiplexer with Local Output</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MUXF8</td>
<td>2-to-1 Lookup Table Multiplexer with General Output</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### Shifters

Shifters are barrel shifters (BRLSHFT) of four and eight bits.

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRLSHFT4</td>
<td>4-Bit Barrel Shifter</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>BRLSHFT8</td>
<td>8-Bit Barrel Shifter</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
</tbody>
</table>
Shift Registers

Shift registers are available in a variety of sizes and capabilities. The naming convention shown in the following figure illustrates available features.

![Shift Register Naming Convention](X4578)

**Shift Register Naming Convention**

<table>
<thead>
<tr>
<th>Design Element</th>
<th>Description</th>
<th>Spartan-II, IIE</th>
<th>Spartan-3</th>
<th>Virtex, E</th>
<th>Virtex II, Pro, Pro X</th>
<th>XC9500/XV/XL</th>
<th>CR XPLA3</th>
<th>CR-II</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SR4CE</strong></td>
<td>4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR8CE</strong></td>
<td>8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR16CE</strong></td>
<td>16-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR4CLE</strong></td>
<td>4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR8CLE</strong></td>
<td>8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR16CLE</strong></td>
<td>16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR4CLED</strong></td>
<td>4-Bit Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR8CLED</strong></td>
<td>8-Bit Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR16CLED</strong></td>
<td>16-Bit Shift Register with Clock Enable and Asynchronous Clear</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR4RE</strong></td>
<td>4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR8RE</strong></td>
<td>8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR16RE</strong></td>
<td>16-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td><strong>SR4RLE</strong></td>
<td>4-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>SR8RLE</td>
<td>8-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>SR16RLE</td>
<td>16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>SR4RLED</td>
<td>4-Bit Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>SR8RLED</td>
<td>8-Bit Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>SR16RLED</td>
<td>16-Bit Shift Register with Clock Enable and Synchronous Reset</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD4CE</td>
<td>4-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD8CE</td>
<td>8-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD16CE</td>
<td>16-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD4CLE</td>
<td>4-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD8CLE</td>
<td>8-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD16CLE</td>
<td>16-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD4CLED</td>
<td>4-Bit Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD8CLED</td>
<td>8-Bit Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD16CLED</td>
<td>16-Bit Dual Edge Triggered Shift Register with Clock Enable and Asynchronous Clear</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD4RE</td>
<td>4-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD8RE</td>
<td>8-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD16RE</td>
<td>16-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD4RLE</td>
<td>4-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>Design Element</td>
<td>Description</td>
<td>Spartan-II, IIE</td>
<td>Spartan-3</td>
<td>Virtex, E</td>
<td>Virtex II, Pro, Pro X</td>
<td>XC9500/XV/XL</td>
<td>CR XPLA3</td>
<td>CR-II</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>SRD8RLE</td>
<td>8-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD16RLE</td>
<td>16-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD4RLED</td>
<td>4-Bit Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD8RLED</td>
<td>8-Bit Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRD16RLED</td>
<td>16-Bit Dual Edge Triggered Shift Register with Clock Enable and Synchronous Reset</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
<tr>
<td>SRL16</td>
<td>16-Bit Shift Register Look-Up-Table (LUT)</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRL16_1</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Negative-Edge Clock</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRL16E</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRL16E_1</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Clock Enable and Negative-Edge Clock and Clock Enable</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRLC16</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Carry</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRLC16_1</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Carry and Negative-Edge Clock</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRLC16E</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Carry and Clock Enable</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRLC16E_1</td>
<td>16-Bit Shift Register Look-Up-Table (LUT) with Carry, Negative-Edge Clock, and Clock Enable</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
**Design Elements**

The remaining sections in this book describe each design element that can be used with the supported architectures.

Design elements are organized in alphanumeric order, with all numeric suffixes in ascending order. For example, FDR precedes FDRS, and ADD4 precedes ADD8, which precedes ADD16.

The following information is provided for each library element, where applicable:

- Graphic symbol
- Applicability table (with primitive versus macro identification)
- Functional description
- Truth table
- Schematic for macros
- VHDL and Verilog instantiation and inference code
- Commonly used constraints
ACC1

1-Bit Loadable Cascadable Accumulator with Carry-In, Carry-Out, and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>ACC1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500VX, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

ACC1 can add or subtract a 1-bit unsigned-binary word to or from the contents of a 1-bit data register and store the results in the register. The register can be loaded with a 1-bit word. The synchronous reset (R) has priority over all other inputs and, when High, causes the output to go to logic level zero during the Low-to-High clock (C) transition. Clock (C) transitions are ignored when clock enable (CE) is Low. The accumulator is asynchronously cleared, outputs Low, when power is applied. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Load

When the load input (L) is High, CE is ignored and the data on the input D0 is loaded into the 1-bit register during the Low-to-High clock (C) transition.

Add

When control inputs ADD and CE are both High, the accumulator adds a 1-bit word (B0) and carry-in (CI) to the contents of the 1-bit register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. The carry-out (CO) is not registered synchronously with the data output. CO always reflects the accumulation of input B0 and the contents of the register, which allows cascading of ACC1s by connecting CO of one stage to CI of the next stage. In add mode, CO acts as a carry-out, and CO and CI are active-High.

Subtract

When ADD is Low and CE is High, the 1-bit word B0 and CI are subtracted from the contents of the register. The result is stored in the register and appears on output Q0 during the Low-to-High clock transition. The carry-out (CO) is not registered synchronously with the data output. CO always reflects the accumulation of input B0 and the contents of the register, which allows cascading of ACC1s by connecting CO of one stage to CI of the next stage. In subtract mode, CO acts as a borrow, and CO and CI are active-Low.
ACC1 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

ACC is schematic and inference only-- not instantiated.

VHDL Inference Code

Following is some "basic" code for inference of the ACC modules.

```vhdl
architecture Behavioral of acc1 is
begin
  process(C, R)
  begin
    if (R = '1') then
      Q <= (others => '0');
    elsif (C'event and C = '1') then
      if (L = '1') then
        Q <= D;
      elsif (CE = '1') then
        if (ADD = '1') then
          Q <= Q + B;
        else
          Q <= Q - B;
        end if;
      end if;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    if (ADD)
      Q <= Q + B;
    else
      Q <= Q - B;
  end
end
```

Commonly Used Constraints

None
ACC4, 8, 16

4-, 8-, 16-Bit Loadable Cascadable Accumulators with Carry-In, Carry-Out, and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>ACC4, ACC8, ACC16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

ACC4, ACC8, ACC16 can add or subtract a 4-, 8-, 16-bit unsigned-binary, respectively or twos-complement word to or from the contents of a 4-, 8-, 16-bit data register and store the results in the register. The register can be loaded with the 4-, 8-, 16-bit word.

The synchronous reset (R) has priority over all other inputs, and when High, causes all outputs to go to logic level zero during the Low-to-High clock (C) transition. Clock (C) transitions are ignored when clock enable (CE) is Low.

The accumulator is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEx, or STARTUP_VIRTEx2 symbol.

Load

When the load input (L) is High, CE is ignored and the data on the D inputs is loaded into the register during the Low-to-High clock (C) transition. ACC4 loads the data on inputs D3 – D0 into the 4-bit register. ACC8 loads the data on D7 – D0 into the 8-bit register. ACC16 loads the data on inputs D15 – D0 into the 16-bit register.

Unsigned Binary Versus Twos Complement

ACC4, ACC8, ACC16 can operate, respectively, on either 4-, 8-, 16-bit unsigned binary numbers or 4-, 8-, 16-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when “overflow” occurs. Unsigned
binary uses CO, while twos complement uses OFL to determine when “overflow” occurs.

**Unsigned Binary Operation**

For unsigned binary operation, ACC4 can represent numbers between 0 and 15, inclusive; ACC8 between 0 and 255, inclusive; and ACC16 between 0 and 65535, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtractor. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds. The carry-out (CO) is not registered synchronously with the data outputs. CO always reflects the accumulation of the B inputs (B3 – B0 for ACC4, B7 – B0 for ACC8, B15 – B0 for ACC16) and the contents of the register. This allows cascading of ACC4s, ACC8s, or ACC16s by connecting CO of one stage to CI of the next stage. An unsigned binary “overflow” that is always active-High can be generated by gating the ADD signal and CO as follows.

\[ \text{unsigned overflow} = \text{CO XOR ADD} \]

Ignore OFL in unsigned binary operation.

**Twos-Complement Operation**

For twos-complement operation, ACC4 can represent numbers between -8 and +7, inclusive; ACC8 between -128 and +127, inclusive; ACC16 between -32768 and +32767, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High. The overflow (OFL) is not registered synchronously with the data outputs. OFL always reflects the accumulation of the B inputs (B3 – B0 for ACC4, B7 – B0 for ACC8, B15 – B0 for ACC16) and the contents of the register, which allows cascading of ACC4s, ACC8s, or ACC16s by connecting OFL of one stage to CI of the next stage.

Ignore CO in twos-complement operation.
ACC8 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
ACC8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
ACC4 Implementation XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II
**Usage**

ACC is schematic and inference only -- not instantiated.

**VHDL Inference Code (ACC4)**

Following is some "basic" code for inference of the ACC modules.

```vhdl
architecture Behavioral of acc4 is
begin
  process (C)
  begin
    if (R = '1') then
      Q <= (others => '0');
    elsif (C'event and C = '1') then
      if (L = '1') then
        Q <= D;
      elsif (CE = '1') then
        if (ADD = '1') then
          Q <= Q + B;
        else
          Q <= Q - B;
        end if;
      end if;
    end if;
  end if;
end process;
```

**ACC8 Implementation XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II**
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    end
  if (ADD)
    Q <= Q + B;
  else
    Q <= Q - B;
end

Commonly Used Constraints
None
ADD1

1-Bit Full Adder with Carry-In and Carry-Out

Architectures Supported

<table>
<thead>
<tr>
<th>ADD1</th>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

ADD1 is a cascadable 1-bit full adder with carry-in and carry-out. It adds two 1-bit words (A and B) and a carry-in (CI), producing a binary sum (S0) output and a carry-out (CO).

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
ADD1 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
This design element is schematic or inference only -- no instantiation.

VHDL Inference Code

```vhdl
architecture Behavioral of ADD is
    signal sum: std_logic_vector(WIDTH downto 0);
    signal zeros: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
begin
    process (CI, A, B, sum)
    begin
        sum <= ('0' & A) + ('0' & B) + (zeros & CI);
        S <= sum(WIDTH-1 downto 0);
        CO <= sum(WIDTH);
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (A or B or CI)
begin
    {CO, S} <= A + B + CI;
end
```

Commonly Used Constraints

None
ADD4, 8, 16

4-, 8-, 16-Bit Cascadable Full Adders with Carry-In, Carry-Out, and Overflow

### Architectures Supported

<table>
<thead>
<tr>
<th>ADD4, ADD8, ADD16</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td></td>
</tr>
</tbody>
</table>

ADD4, ADD8, and ADD16 add two words and a carry-in (CI), producing a sum output and carry-out (CO) or overflow (OFL). ADD4 adds $A_3 - A_0$, $B_3 - B_0$, and CI producing the sum output $S_3 - S_0$ and CO (or OFL). ADD8 adds $A_7 - A_0$, $B_7 - B_0$, and CI, producing the sum output $S_7 - S_0$ and CO (or OFL). ADD16 adds $A_{15} - A_0$, $B_{15} - B_0$ and CI, producing the sum output $S_{15} - S_0$ and CO (or OFL).

### Unsigned Binary Versus Twos Complement

ADD4, ADD8, ADD16 can operate on either 4-, 8-, 16-bit unsigned binary numbers or 4-, 8-, 16-bit twos-complement numbers, respectively. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when “overflow” occurs. Unsigned binary uses CO, while twos-complement uses OFL to determine when “overflow” occurs. To interpret the inputs as unsigned binary, follow the CO output. To interpret the inputs as twos complement, follow the OFL output.

### Unsigned Binary Operation

For unsigned binary operation, ADD4 can represent numbers between 0 and 15, inclusive; ADD8 between 0 and 255, inclusive; ADD16 between 0 and 65535, inclusive. CO is active (High) when the sum exceeds the bounds of the adder.

OFL is ignored in unsigned binary operation.

### Twos-Complement Operation

For twos-complement operation, ADD4 can represent numbers between -8 and +7, inclusive; ADD8 between -128 and +127, inclusive; ADD16 between -32768 and +32767, inclusive. OFL is active (High) when the sum exceeds the bounds of the adder. CO is ignored in twos-complement operation.
ADD8 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
ADD8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
ADD4 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
ADD8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This design element is schematic or inference only -- no instantiation.

VHDL Inference Code (ADD4)

```vhdl
architecture Behavioral of ADD is

    signal sum: std_logic_vector(WIDTH-1 downto 0);
    signal zeros: std_logic_vector(WIDTH-1 downto 0) := (others => '0');

begin

    process (CI, A, B, sum)
    begin
        sum <= ('0' & A) + ('0' & B) + (zeros & CI);
        S <= sum(WIDTH-1 downto 0);
        CO <= sum(WIDTH);
    end process;

end Behavioral;
```

Verilog Inference Code (ADD4)

```verilog
always @ (A or B or CI)
begin
    {CO,sum} <= A + B + CI;
end
```
Commonly Used Constraints

None
1-Bit Cascadable Adder/Subtracter with Carry-In and Carry-Out

**Architectures Supported**

<table>
<thead>
<tr>
<th>ADSU1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

When the ADD input is High, two 1-bit words (A0 and B0) are added with a carry-in (CI), producing a 1-bit output (S0) and a carry-out (CO). When the ADD input is Low, B0 is subtracted from A0, producing a result (S0) and borrow (CO). In add mode, CO represents a carry-out, and CO and CI are active-High. In subtract mode, CO represents a borrow, and CO and CI are active-Low.

**Add Function, ADD=1**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Subtract Function, ADD=0**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Subtract Function, ADD=0

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>B0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of adsu4 is
begin
```
process (A0,ADD,B0)
begin
  if (ADD='1') then
    S <= A0 + B0;
  else
    S <= A0 - B0;
  end if;
end process;
end Behavioral;

Verilog Inference Code
always @(A0 or ADD or B0)
begin
  if (ADD)
    S0 <= A0 + B0;
  else
    S0 <= A0 - B0;
end

VHDL Instantiation Template
-- Component Declaration for ADSU1 should be placed
-- after architecture statement but before begin keyword

component ADSU1
  port (CO : out STD_ULOGIC;
        S0 : out STD_ULOGIC;
        A0 : in STD_ULOGIC;
        ADD: in STD_ULOGIC;
        B0 : in STD_ULOGIC;
        CI : in STD_ULOGIC);
end component;

-- Component Attribute specification for ADSU1
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes here

-- Component Instantiation for ADSU1 should be placed
-- in architecture after the begin keyword

ADSU1_INSTANCE_NAME : ADSU1
  -- synthesis translate_off
generic map (CDS_ACTION => "string_value")
  -- synthesis translate_on
port map (CO => user_CO,
         S0 => user_SO,
         A0 => user_A0,
         ADD => user_ADD,
         B0 => user_B0,
         CI => user_CI);

Verilog Instantiation Template
ADSU1 instance_name (.CO (user_CO),
.S0 (user_S0),
.A0 (user_A0),
.ADD(user_ADD),
.B0 (user_B0),
.CI (user_CI));

Commonly Used Constraints

None
ADSU4, 8, 16

4-, 8-, 16-Bit Cascadable Adders/Subtracters with Carry-In, Carry-Out, and Overflow

Architectures Supported

<table>
<thead>
<tr>
<th>ADSU4, ADSU8, ADSU16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

When the ADD input is High, ADSU4, ADSU8, and ADSU16 add two words and a carry-in (CI), producing a sum output and carry-out (CO) or overflow (OFL). ADSU4 adds two 4-bit words (A3 – A0 and B3 – B0) and a CI, producing a 4-bit sum output (S3 – S0) and CO or OFL. ADSU8 adds two 8-bit words (A7 – A0 and B7 – B0) and a CI producing, an 8-bit sum output (S7 – S0) and CO or OFL. ADSU16 adds two 16-bit words (A15 – A0 and B15 – B0) and a CI, producing a 16-bit sum output (S15 – S0) and CO or OFL.

When the ADD input is Low, ADSU4, ADSU8, and ADSU16 subtract Bz – B0 from Az– A0, producing a difference output and CO or OFL. ADSU4 subtracts B3 – B0 from A3 – A0, producing a 4-bit difference (S3 – S0) and CO or OFL. ADSU8 subtracts B7 – B0 from A7 – A0, producing an 8-bit difference (S7 – S0) and CO or OFL. ADSU16 subtracts B15 – B0 from A15 – A0, producing a 16-bit difference (S15 – S0) and CO or OFL.

In add mode, CO and CI are active-High. In subtract mode, CO and CI are active-Low. OFL is active-High in add and subtract modes.

ADSU4, ADSU8, and ADSU16 CI and CO pins do not use the CPLD carry chain.

Unsigned Binary Versus Twos Complement

ADSU4, ADSU8, ADSU16 can operate, respectively, on either 4-, 8-, 16-bit unsigned binary numbers or 4-, 8-, 16-bit twos-complement numbers. If the inputs are interpreted as unsigned binary, the result can be interpreted as unsigned binary. If the inputs are interpreted as twos complement, the output can be interpreted as twos complement. The only functional difference between an unsigned binary operation and a twos-complement operation is how they determine when “overflow” occurs. Unsigned binary uses CO, while twos complement uses OFL to determine when “overflow” occurs.

With adder/subtracters, either unsigned binary or twos-complement operations cause an overflow. If the result crosses the overflow boundary, an overflow is generated. Similarly, when the result crosses the carry-out boundary, a carry-out is generated. The following figure shows the ADSU carry-out and overflow boundaries.
ADSU Carry-Out and Overflow Boundaries

Unsigned Binary Operation

For unsigned binary operation, ADSU4 can represent numbers between 0 and 15, inclusive; ADSU8 between 0 and 255, inclusive; ADSU16 between 0 and 65535, inclusive. In add mode, CO is active (High) when the sum exceeds the bounds of the adder/subtractor. In subtract mode, CO is an active-Low borrow-out and goes Low when the difference exceeds the bounds.

An unsigned binary “overflow” that is always active-High can be generated by gating the ADD signal and CO as follows.

\[
\text{unsigned overflow} = \text{CO XOR ADD}
\]

OFL is ignored in unsigned binary operation.

Twos-Complement Operation

For twos-complement operation, ADSU4 can represent numbers between -8 and +7, inclusive; ADSU8 between -128 and +127, inclusive; ADSU16 between -32768 and +32767, inclusive. If an addition or subtraction operation result exceeds this range, the OFL output goes High.

CO is ignored in twos-complement operation.
ADSU8 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
ADSU8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
ADSU4 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

ADSU8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usability

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of adsu4 is

begin

process (A,ADD,B)
begin
    if (ADD='1') then
        S <= A + B;
    else
        S <= A - B;
    end if;
end process;

end Behavioral;

Verilog Inference Code

always @ (A or ADD or B)
begin
    if (ADD)
        S <= A + B;
    else
        S <= A - B;
end
AND2-9

2- to 9-Input AND Gates with Inverted and Non-Inverted Inputs

Architectures Supported

<table>
<thead>
<tr>
<th>AND2, AND3, AND4, AND5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND2B1, AND2B2, AND4B1, AND4B2, AND4B3, AND4B4, AND5B1, AND5B2, AND5B3, AND5B4, AND5B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND3B1, AND3B2, AND3B3,</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND6, AND7, AND8, AND9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>
AND Gate Representations

AND functions of up to five inputs are available in any combination of inverting and non-inverting inputs. AND functions of six to nine inputs are available with only non-inverting inputs. To make some or all inputs inverting, use external inverters. Because each input uses a CLB resource in Spartan-II, Spartan-3, Spartan-IIIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X replace functions with unused inputs with functions having the appropriate number of inputs.

See “AND12, 16” for information on additional AND functions for Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, and Spartan-3.
AND8 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

AND8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

If possible, it is recommended that these design elements be inferred rather than instantiated.

VHDL Inference Code

AND2 example:

```vhdl
process (I0, I1)
begin
  O <= I0 and I1;
end process;
```

AND5B3 example:

```vhdl
process (I0, I1, I2, I3, I4)
begin
  O <= (not I0) and (not I1) and (not I2) and I3 and I4;
end process;
```

Verilog Inference Code

AND2:

```verilog
always @ (I0 or I1)
begin
  O <= I0 && I1;
end
```

AND5B3:

```verilog
always @ (I0 or I1 or I2 or I3 or I4)
begin
  O <= !I0 && !I1 && !I2 && I3 && I4;
end
```

VHDL Instantiation Template for AND2, AND2B1, or AND2B2

```
-- Component Declaration for AND2, AND2B1, or AND2B2 should
-- be placed after architecture statement but before begin
-- keyword
component {AND2|AND2B1|AND2B2}
  port (O : out STD_ULOGIC;
        I1 : in STD_ULOGIC;
        I0 : in STD_ULOGIC);
end component;

-- Component Attribute specification for AND2, AND2B1, or
-- AND2B2 should be placed after architecture declaration
-- but before the begin keyword
-- Enter attributes here

-- Component Instantiation for AND2, AND2B1, or AND2B2
-- should be placed in architecture after the begin
-- keyword
```
 INSTANCE_NAME : (AND2|AND2B1|AND2B2)
    port map (O => user_O,
    I0 => user_I0,
    I1 => user_I1);

Verilog Instantiation Template for AND2, AND2B1, or AND2B2

ANDn instance_name (.O (user_O),
                     .I0 (user_I0),
                     .I1 (user_I1));

VHDL Instantiation Template for AND3 Through AND3B3

-- Component Declaration for AND3 through AND3B3 should
-- be placed after architecture statement but before begin
-- keyword

component (AND3|AND3B1|AND3B2|AND3B3)
    port (O : out STD_ULOGIC;
          I0 : in STD_ULOGIC;
          I1 : in STD_ULOGIC;
          I2 : in STD_ULOGIC);
end component;

-- Component Attribute specification for AND3 through AND3B3
-- should be placed after architecture declaration
-- but before the begin keyword

-- Enter attributes here

-- Component Instantiation for AND3 through AND3B3
-- should be placed in architecture after the begin
-- keyword

 INSTANCE_NAME : (AND3|AND3B1|AND3B2|AND3B3)
    port map (O => user_O,
    I0 => user_I0,
    I1 => user_I1,
    I2 => user_I2);

Verilog Instantiation Template for AND3 Through AND3B3

ANDn instance_name (.O (user_O),
                     .I0 (user_I0),
                     .I1 (user_I1),
                     .I2 (user_I2));
VHDL Instantiation Template for AND4 Through AND4B4

-- Component Declaration for AND4 through AND4B4 should
-- be placed after architecture statement but before begin
-- keyword

component {AND4|AND4B1|AND4B2|AND4B3|AND4B4}
  port (O : out STD_ULOGIC;
       I0 : in STD_ULOGIC;
       I1 : in STD_ULOGIC;
       I2 : in STD_ULOGIC;
       I3 : in STD_ULOGIC);
end component;

-- Component Attribute specification for AND4 through AND4B4
-- should be placed after architecture declaration
-- but before the begin keyword

-- Enter attributes here

-- Component Instantiation for AND4 through AND4B4
-- should be placed in architecture after the begin
-- keyword

INSTANCE_NAME : AND4_thru_AND4B4
  port map (O => user_O,
           I0 => user_I0,
           I1 => user_I1,
           I2 => user_I2,
           I3 => user_I3);

Verilog Instantiation Template for AND4 through AND4B4

ANDn instance_name (.O (user_O),
           .I0 (user_I0),
           .I1 (user_I1),
           .I2 (user_I2),
           .I3 (user_I3));
VHDL Instantiation Template for AND5 Through AND5B5

-- Component Declaration for AND5 through AND5B5 should
-- be placed after architecture statement but before begin
-- keyword

component {AND5|AND5B1|AND5B2|AND5B3|AND5B4|AND5B5}
port (O : out STD_ULOGIC;
     I0 : in STD_ULOGIC;
     I1 : in STD_ULOGIC;
     I2 : in STD_ULOGIC;
     I3 : in STD_ULOGIC;
     I4 : in STD_ULOGIC);
end component;

-- Component Attribute specification for AND5 through AND5B5
-- should be placed after architecture declaration
-- but before the begin keyword

-- Enter attributes here

-- Component Instantiation for AND5 through AND5B5
-- should be placed in architecture after the begin
-- keyword

INSTANCE_NAME : AND5_thru_AND5B5
port map (O => user_O,
          I0 => user_I0,
          I1 => user_I1,
          I2 => user_I2,
          I3 => user_I3,
          I4 => user_I4);

Verilog Instantiation Template for AND5 through AND5B5

ANDn instance_name (.O (user_O),
                    .I0 (user_I0),
                    .I1 (user_I1),
                    .I2 (user_I2),
                    .I3 (user_I3),
                    .I4 (user_I4));

Commonly Used Constraints

None
AND12, 16

12- and 16-Input AND Gates with Non-Inverted Inputs

Architectures Supported

<table>
<thead>
<tr>
<th>Function</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND12</td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>AND12</td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>AND12</td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>AND12</td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>AND16</td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>AND16</td>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

AND12 and AND16 functions are performed in the Configurable Logic Block (CLB) function generator.

The 12- and 16-input AND functions are available only with non-inverting inputs. To invert all of some inputs, use external inverters.

See “AND2-9” for information on more AND functions.
AND12, 16

AND12 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

AND16 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
AND16 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage
For HDL, it is recommended that these design elements be inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of and12 is
begin
process (I0, I1, I2, I3, I4, I5, I6, I7, I8, I9, I10, I11)
begin
  O <= I0 and I1 and I2 and I3 and I4 and I5 and I6 and I7
  and I8 and I9 and I10 and I11;
end process;

end Behavioral;
Verilog Inference Code

always @ (I0 or I1 or I2 or I3 or I4 or I5 or I6 or I7 or I8 or I9 or I10 or I11)
begin
  O <= I0 & I1 & I2 & I3 & I4 & I5 & I6 & I7 & I8 & I9 & I10 & I11;
end

Commonly Used Constraints

None
BRLSHFT4, 8

4-, 8-Bit Barrel Shifters

Architectures Supported

<table>
<thead>
<tr>
<th>BRLSHFT4, BRLSHFT8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Macro</td>
</tr>
</tbody>
</table>

BRLSHFT4, a 4-bit barrel shifter, can rotate four inputs (I3 – I0) up to four places. The control inputs (S1 and S0) determine the number of positions, from one to four, that the data is rotated. The four outputs (O3 – O0) reflect the shifted data inputs.

BRLSHFT8, an 8-bit barrel shifter, can rotate the eight inputs (I7 – I0) up to eight places. The control inputs (S2 – S0) determine the number of positions, from one to eight, that the data is rotated. The eight outputs (O7 – O0) reflect the shifted data inputs.

BRLSHFT4 Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>S0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

BRLSHFT8 Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>S1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### BRLSHFT8 Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>S1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Usage
For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of brlshft4 is
begin
process (I,S)
begin
  case S is
    when "00" =>
      O <= I;
    when "01" =>
      O(3) <= I(0);
      O(2 downto 0) <= I(3 downto 1);
    when "10" =>
      O(3 downto 2) <= I(1 downto 0);
      O(1 downto 0) <= I(3 downto 2);
    when "11" =>
      O(3 downto 1) <= I(2 downto 0);
      O(0) <= I(3);
    when others =>
      O <= I;
  end case;
end process;
end Behavioral;

Verilog Inference Code

always @(I or S)
begin
  case (S)
    2'b00 : O <= I;
    2'b01 : O <= {I[0],I[3:1]};
    2'b10 : O <= {I[1:0],I[3:2]};
    2'b11 : O <= {I[2:0],I[3]};
  endcase
end
Spartan-II Boundary Scan Logic Control Circuit

Architectures Supported

<table>
<thead>
<tr>
<th>BSCAN_SPARTAN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

* Supported for Spartan-II, but not for Spartan-IIIE, which is supported by BSCAN_VIRTEX.

The BSCAN_SPARTAN2 symbol creates internal boundary scan chains in a Spartan-II device. The 4-pin JTAG interface (TDI, TDO, TCK, and TMS) are dedicated pins in Spartan-II. To use normal JTAG for boundary scan purposes, just hook up the JTAG pins to the port and go. The pins on the BSCAN_SPARTAN2 symbol do not need to be connected, unless those special functions are needed to drive an internal scan chain.

A signal on the TDO1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL1 output goes High to indicate that the USER1 instruction is active. The DRCK1 output provides USER1 access to the data register clock (generated by the TAP controller). The TDO2 and SEL2 pins perform a similar function for the USER2 instruction and the DRCK2 output provides USER2 access to the data register clock (generated by the TAP controller). The RESET, UPDATE, and SHIFT pins represent the decoding of the corresponding state of the boundary scan internal state machine. The TDI pin provides access to the TDI signal of the JTAG port in order to shift data into an internal scan chain.

Note: For specific information on boundary scan for an architecture, see The Programmable Logic Data Book.

Usage

This design element is supported for instantiation and schematics but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for BSCAN_SPARTAN2 should be placed
-- after architecture statement but before begin keyword

component BSCAN_SPARTAN2
  port (DRCK1 : out STD_ULOGIC;
       DRCK2 : out STD_ULOGIC;
       RESET : out STD_ULOGIC;
       SEL1 : out STD_ULOGIC;
```

---
SEL2 : out STD_ULOGIC;
SHIFT : out STD_ULOGIC;
TDI : out STD_ULOGIC;
UPDATE : out STD_ULOGIC;
TD01 : in STD_ULOGIC;
TD02 : in STD_ULOGIC);
end component;

-- Component Attribute specification for BSCAN_SPARTAN2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BSCAN_SPARTAN2 should be
-- placed in architecture after the begin keyword
BSCAN_SPARTAN2_INSTANCE_NAME : BSCAN_SPARTAN2
  port map (DRCK1 => user_DRCK1,
            DRCK2 => user_DRCK2,
            RESET => user_RESET,
            SEL1 => user_SEL1,
            SEL2 => user_SEL2,
            SHIFT => user_SHIFT,
            TDI => user_TDI,
            UPDATE => user_UPDATE,
            TD01 => user_TD01,
            TD02 => user_TD02);

Verilog Instantiation Template
BSCAN_SPARTAN2 instance_name (.DRCK1 (user_DRCK1),
                              .DRCK2 (user_DRCK2),
                              .RESET (user_RESET),
                              .SEL1 (user_SEL1),
                              .SEL2 (user_SEL2),
                              .SHIFT (user_SHIFT),
                              .TDI (user_TDI),
                              .UPDATE (user_UPDATE),
                              .TD01 (user_TD01),
                              .TD02 (user_TD02));

Commonly Used Constraints
None
**BSCAN_SPARTAN3**

**Spartan-3 Boundary Scan Logic Control Circuit**

**Architectures Supported**

<table>
<thead>
<tr>
<th>BSCAN_SPARTAN3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BSCAN_SPARTAN3 provides access to the BSCAN sites on a Spartan-3 device. It is used to create internal boundary scan chains. The 4-pin JTAG interface (TDI, TDO, TCK, and TMS) are dedicated pins in Spartan-3. To use normal JTAG for boundary scan purposes, just hook up the JTAG pins to the port and go. The pins on the BSCAN_SPARTAN3 symbol do not need to be connected, unless those special functions are needed to drive an internal scan chain.

A signal on the TDO1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL1 output goes High to indicate that the USER1 instruction is active. The DRCK1 output provides USER1 access to the data register clock (generated by the TAP controller). The TDO2 and SEL2 pins perform a similar function for the USER2 instruction and the DRCK2 output provides USER2 access to the data register clock (generated by the TAP controller). The RESET, UPDATE, SHIFT, and CAPTURE pins represent the decoding of the corresponding state of the boundary scan internal state machine. The TDI pin provides access to the TDI signal of the JTAG port in order to shift data into an internal scan chain.

**Usage**

This design element is instantiated rather than inferred.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for BSCAN_SPARTAN3 should be placed
-- after architecture statement but before begin keyword

component BSCAN_SPARTAN3
    port (CAPTURE : out STD_ULOGIC;
          DRCK1 : out STD_ULOGIC;
          DRCK2 : out STD_ULOGIC;
          RESET : out STD_ULOGIC;
          SEL1 : out STD_ULOGIC;
          SEL2 : out STD_ULOGIC;
          SHIFT : out STD_ULOGIC;
          TDI : out STD_ULOGIC;
          UPDATE : out STD_ULOGIC;
          TD01 : in STD_ULOGIC;
```
TD02 : in STD_ULOGIC);
end component;

-- Component Attribute specification for BSCAN_SPARTAN3
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BSCAN_SPARTAN3 should be
-- placed in architecture after the begin keyword

BSCAN_SPARTAN3_INSTANCE_NAME : BSCAN_SPARTAN3
port map (CAPTURE => user_CAPTURE,
          DRCK1 => user_DRCK1,
          DRCK2 => user_DRCK2,
          RESET => user_RESET,
          SEL1 => user_SEL1,
          SEL2 => user_SEL2,
          SHIFT => user_SHIFT,
          TDI => user_TDI,
          UPDATE => user_UPDATE,
          TD01 => user_TDO1,
          TD02 => user_TDO2);

Verilog Instantiation Template

BSCAN_SPARTAN3 instance_name (.CAPTURE (user_CAPTURE),
                              .DRCK1 (user_DRCK1),
                              .DRCK2 (user_DRCK2),
                              .RESET (user_RESET),
                              .SEL1 (user_SEL1),
                              .SEL2 (user_SEL2),
                              .SHIFT (user_SHIFT),
                              .TDI (user_TDI),
                              .UPDATE (user_UPDATE),
                              .TDO1 (user_TDO1),
                              .TDO2 (user_TDO2));

Commonly Used Constraints

None.
BSCAN_VIRTEX

Virtex Boundary Scan Logic Control Circuit

Architectures Supported

<table>
<thead>
<tr>
<th>BSCAN_VIRTEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

*Supported for Spartan-II, but not for Spartan-II, which is supported by BSCAN_SPARTAN2.

The BSCAN_VIRTEX symbol is used to create internal boundary scan chains in a Virtex or Virtex-E device. The 4-pin JTAG interface (TDI, TDO, TCK, and TMS) are dedicated pins in Virtex and Virtex-E. To use normal JTAG for boundary scan purposes, just hook up the JTAG pins to the port and go. The pins on the BSCAN_VIRTEX symbol do not need to be connected, unless those special functions are needed to drive an internal scan chain.

Note: For Virtex-II, Virtex-II Pro, and Virtex-II Pro X, see “BSCAN_VIRTEX2”.

A signal on the TDO1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL1 output goes High to indicate that the USER1 instruction is active. The DRCK1 output provides USER1 access to the data register clock (generated by the TAP controller). The TDO2 and SEL2 pins perform a similar function for the USER2 instruction and the DRCK2 output provides USER2 access to the data register clock (generated by the TAP controller). The RESET, UPDATE, and SHIFT pins represent the decoding of the corresponding state of the boundary scan internal state machine. The TDI pin provides access to the TDI signal of the JTAG port in order to shift data into an internal scan chain.

Note: For specific information on boundary scan for an architecture, see The Programmable Logic Data Book.

Usage

This design element is supported for instantiation and schematics but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for BSCAN_VIRTEX should be placed
-- after architecture statement but before begin keyword

component BSCAN_VIRTEX
    port (DRCK1 : out STD_ULOGIC;
          DRCK2 : out STD_ULOGIC;
```
RESET : out STD_ULOGIC;
SEL1 : out STD_ULOGIC;
SEL2 : out STD_ULOGIC;
SHIFT : out STD_ULOGIC;
TDI : out STD_ULOGIC;
UPDATE : out STD_ULOGIC;
TD01 : in STD_ULOGIC;
TD02 : in STD_ULOGIC);
end component;

-- Component Attribute specification for BSCAN_VIRTEX
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BSCAN_VIRTEX should be
-- placed in architecture after the begin keyword

BSCAN_VIRTEX_INSTANCE_NAME : BSCAN_VIRTEX
port map (DRCK1 => user_DRCK1,
          DRCK2 => user_DRCK2,
          RESET => user_RESET,
          SEL1 => user_SEL1,
          SEL2 => user_SEL2,
          SHIFT => user_SHIFT,
          TDI => user_TDI,
          UPDATE => user_UPDATE,
          TD01 => user_TD01,
          TD02 => user_TD02);

Verilog Instantiation Template

BSCAN_VIRTEX instance_name (.DRCK1 (user_DRCK1),
                           .DRCK2 (user_DRCK2),
                           .RESET (user_RESET),
                           .SEL1 (user_SEL1),
                           .SEL2 (user_SEL2),
                           .SHIFT (user SHIFT),
                           .TDI (user_TDI),
                           .UPDATE (user_UPDATE),
                           .TD01 (user_TD01),
                           .TD02 (user_TD02));

Commonly Used Constraints
None
BSCAN_VIRTEX2

Virtex-II Boundary Scan Logic Control Circuit

Architectures Supported

<table>
<thead>
<tr>
<th>BSCAN_VIRTEX2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BSCAN_VIRTEX2 provides access to the BSCAN sites on a Virtex-II, Virtex-II Pro, or Virtex-II Pro X device. It is used to create internal boundary scan chains. The 4-pin JTAG interface (TDI, TDO, TCK, and TMS) are dedicated pins in Virtex-II, Virtex-II Pro, and Virtex-II Pro X. To use normal JTAG for boundary scan purposes, just hook up the JTAG pins to the port and go. The pins on the BSCAN_VIRTEX2 symbol do not need to be connected, unless those special functions are needed to drive an internal scan chain.

Note: For Virtex and Virtex-E, see "BSCAN_VIRTEX".

A signal on the TDO1 input is passed to the external TDO output when the USER1 instruction is executed; the SEL1 output goes High to indicate that the USER1 instruction is active. The DRCK1 output provides USER1 access to the data register clock (generated by the TAP controller). The TDO2 and SEL2 pins perform a similar function for the USER2 instruction and the DRCK2 output provides USER2 access to the data register clock (generated by the TAP controller). The RESET, UPDATE, SHIFT, and CAPTURE pins represent the decoding of the corresponding state of the boundary scan internal state machine. The TDI pin provides access to the TDI signal of the JTAG port in order to shift data into an internal scan chain.

Note: For specific information on boundary scan for an architecture, see The Programmable Logic Data Book.

Usage

This design element is supported for instantiation and schematics but not for inference.

VHDL Instantiation Template

```
-- Component Declaration for BSCAN_VIRTEX2 should be placed
-- after architecture statement but before begin keyword

component BSCAN_VIRTEX2
  port (CAPTURE : out STD_ULOGIC;
         DRCK1 : out STD_ULOGIC;
         DRCK2 : out STD_ULOGIC;
         RESET : out STD_ULOGIC;
```

SEL1 : out STD_ULOGIC;
SEL2 : out STD_ULOGIC;
SHIFT : out STD_ULOGIC;
TDI : out STD_ULOGIC;
UPDATE : out STD_ULOGIC;
TD01 : in STD_ULOGIC;
TD02 : in STD_ULOGIC);

end component;

-- Component Attribute specification for BSCAN_VIRTEX2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BSCAN_VIRTEX2 should be
-- placed in architecture after the begin keyword

BSCAN_VIRTEX2_INSTANCE_NAME : BSCAN_VIRTEX2
port map (CAPTURE =>'user_CAPTURE,
DRCK1 => 'user_DRCK1,
DRCK2 => 'user_DRCK2,
RESET => 'user_RESET,
SEL1 => 'user_SEL1,
SEL2 => 'user_SEL2,
SHIFT => 'user_SHIFT,
TDI => 'user_TDI,
UPDATE => 'user_UPDATE,
TD01 => 'user_TD01,
TD02 => 'user_TD02);

Verilog Instantiation Template

BSCAN_VIRTEX2 instance_name (.CAPTURE (user_CAPTURE),
  .DRCK1 (user_DRCK1),
  .DRCK2 (user_DRCK2),
  .RESET (user_RESET),
  .SEL1 (user_SEL1),
  .SEL2 (user_SEL2),
  .SHIFT (user_SHIFT),
  .TDI (user_TDI),
  .UPDATE (user_UPDATE),
  .TD01 (user_TD01),
  .TD02 (user_TD02));

Commonly Used Constraints

None
General-Purpose Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>BUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro,</td>
</tr>
<tr>
<td>Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

BUF is a general purpose, non-inverting buffer.

In Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, BUF is usually not necessary and is removed by the partitioning software (MAP).

In XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, BUF is usually removed, unless you inhibit optimization by applying the OPT=OFF attribute to the BUF symbol.

Usage

This design is supported in schematics and instantiation but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for BUF should be placed
-- after architecture statement but before begin keyword

component BUF
    port (O : out STD_ULOGIC;
        I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUF
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUF should be placed
-- in architecture after the begin keyword

BUF_INSTANCE_NAME : BUF
    port map (O => user_O,
               I => user_I);
```
Verilog Instantiation Template

BUF instance_name (.O (user_O),
    .I (user_I));

Commonly Used Constraints

None
BUF4, BUF8, BUF16

General-Purpose Buffers

Architectures Supported

<table>
<thead>
<tr>
<th>BUF4, BUF8, BUF16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

BUF4, 8, 16 are general purpose, non-inverting buffers.

In XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, BUF4, BUF8, and BUF16 are usually removed, unless you inhibit optimization by applying the OPT=OFF attribute to the BUF4, BUF8, or BUF16 symbol or by using the LOGIC_OPT=OFF global attribute.

BUF8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

These design elements are schematic only.
BUFCF

Fast Connect Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>BUFCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BUFCF is a single fast connect buffer used to connect the outputs of the LUTs and some dedicated logic directly to the input of another LUT. Using this buffer implies CLB packing. No more than four LUTs may be connected together as a group.

Usage

This design element is supported for schematics and instantiation but not for inference.

VHDL Instantiation Template

```
-- Component Declaration for BUFCF should be placed
-- after architecture statement but before begin keyword

component BUFCF
    port (O : out STD_ULOGIC;
           I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFCF
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFCF should be placed
-- in architecture after the begin keyword

BUFCF_INSTANCE_NAME : BUFCF
    port map (O => user_O,
              I => user_I);
```

Architectures Supported

- Spartan-II, Spartan-IIIE: Primitive
- Spartan-3: Primitive
- Virtex, Virtex-E: Primitive
- Virtex-II, Virtex-II Pro, Virtex-II Pro X: Primitive
- XC9500, XC9500XV, XC9500XL: No
- CoolRunner XPLA3: No
- CoolRunner-II: No
Verilog Instantiation Template

BUFCF instance_name (.O (user_0),
                   .I (user_I));

Commonly Used Constraints

None
Internal 3-State Buffers with Active High Enable

Architectures Supported

<table>
<thead>
<tr>
<th>BUFE</th>
<th>BUFE4, BUFE8, BUFE16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BUFE, BUFE4, BUFE8, and BUFE16 are single or multiple 3-state buffers with inputs I, I3 – I0, I7 – I0, and I15 – I0, respectively; outputs O, O3 – O0, O7 – O0, and O15 – O0, respectively; and active-High output enable (E). When E is High, data on the inputs of the buffers is transferred to the corresponding outputs. When E is Low, the output is high impedance (Z state or Off). The outputs of the buffers are connected to horizontal longlines in FPGA architectures.

The outputs of separate BUFE symbols can be tied together to form a bus or a multiplexer. Make sure that only one E is High at any one time. If none of the E inputs is active-High, a “weak-keeper” circuit (Spartan-II, Spartan-IIIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X) keeps the output bus from floating but does not guarantee that the bus remains at the last value driven onto it.

For XC9500 devices, BUFE output nets assume the High logic level when all connected BUFE/BUFT buffers are disabled.

For Spartan-II, Spartan-IIIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, BUFE elements need a PULLUP element connected to their output. NGDBuild inserts a PULLUP element if one is not connected.

Usage

These design elements are supported for schematic, inference, and instantiation.

VHDL Inference Code

```vhdl
architecture Behavioral of bufe is
begin
process (I, E)
begin
if (E = '1') then
    O <= I;
else
    O <= 'Z';
end if;
end process;
end Behavioral;
```
Verilog Inference Code

always @ (I or E)
begin
  if (E)
    O <= I;
  else
    O <= 1'bZ;
end

VHDL Instantiation Template

-- Component Declaration for BUFE should be placed
-- after architecture statement but before begin keyword

component BUFE
  port (O : out STD_ULOGIC;
       E : in STD_ULOGIC;
       I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFE
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes here

-- Component Instantiation for BUFE should be placed
-- in architecture after the begin keyword

BUFE_INSTANCE_NAME : BUFE
  port map (O => user_O,
            E => user_E,
            I => user_I);

Verilog Instantiation Template

BUFE instance_name (.O (user_O),
                        .E (user_E),
                        .I (user_I));

Commonly Used Constraints

None
Global Clock Buffer

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>BUFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

BUFG, an architecture-independent global buffer, distributes high fan-out clock signals throughout a PLD device. The Xilinx implementation software converts each BUFG to an appropriate type of global buffer for the target PLD device. To use a specific type of buffer, instantiate it manually.

To use a BUFG in a schematic, connect the input of the BUFG symbol to the clock source. Depending on the target PLD family, the clock source can be an external PAD symbol, an IBUF symbol, or internal logic. For a negative-edge clock input, insert an INV (inverter) symbol between the BUFG output and the clock input. The inversion is implemented at the Configurable Logic Block (CLB) or Input Output Block (IOB) clock pin.

XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II

Consult the device data sheet for the number of available global pins. For these architectures BUFG is always implemented using an IOB. Connect the input of BUFG to an IPAD or an IOPAD that represents an external signal source. Each BUFG can drive any number of register clocks in a design. The output of a BUFG may also be used as an ordinary input signal to other logic elsewhere in the design.

Virtex, Virtex-E, Spartan-II, Spartan-IIE

In Virtex, Virtex-E, Spartan-II, and Spartan-IIE, the BUFG cannot be driven directly from a pad. It can be driven from an IBUFG to indicate to use the dedicated pin (GCLKIOB pin) or from an internal driver to create an internal clock. BUFG can also be driven with an IBUF to represent an externally driven clock that does not use the dedicated pin.

Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, clock buffers are multiplexed clock buffers. In Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, a BUFG is implemented using a BUFGMUX with the S_B input tied high, basically meaning the S input is tied low. I1 is unused. I0 is used.
Usage

This design element is supported for schematic and instantiation. Synthesis tools usually infer a BUFGP on any clock net. If there are more clock nets than BUFGPs, the synthesis tool usually instantiates BUFGPs for the clocks that are most utilized. The BUFGP contains both a BUFG and an IBUFG.

VHDL Instantiation Template

-- Component Declaration for BUFG should be placed
-- after architecture statement but before begin keyword

component BUFG
    port (O : out STD_ULOGIC;
          I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFG
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFG should be placed
-- in architecture after the begin keyword

BUFG_INSTANCE_NAME : BUFG
    port map (O => user_O,
              I => user_I);

Verilog Instantiation Template

BUFG instance_name (.O (user_O),
                    .I (user_I));

Commonly Used Constraints
LOC
BUFGCE

Global Clock MUX Buffer with Clock Enable and Output State 0

Architectures Supported

<table>
<thead>
<tr>
<th>BUFGCE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BUFGCE is a global clock buffer with a single gated input. Its O output is "0" when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output.

<table>
<thead>
<tr>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
</tr>
</tbody>
</table>

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for BUFGCE should be placed
-- after architecture statement but before begin keyword

component BUFGCE
  port (O : out STD_ULOGIC;
       CE : in STD_ULOGIC;
       I : in STD_ULOGIC);
end component;
```

X9307

X9384
-- Component Attribute specification for BUFGCE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFGCE should be placed
-- in architecture after the begin keyword

BUFGCE_INSTANCE_NAME : BUFGCE
    port map (O => user_O,
              CE => user_CE,
              I => user_I);

Verilog Instantiation Template

BUFGCE instance_name (.O (user_O),
                       .CE (user_CE),
                       .I (user_I));

Commonly Used Constraints

LOC
BUFGCE_1

Global Clock MUX Buffer with Clock Enable and Output State 1

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BUFGCE_1 is a multiplexed global clock buffer with a single gated input. Its O output is High (1) when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>CE</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

BUFGCE_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for BUFGCE_1 should be placed
-- after architecture statement but before begin keyword

component BUFGCE_1
    port (O : out STD_ULOGIC;
          CE : in STD_ULOGIC;
          I: in STD_ULOGIC);
end component;
```
-- Component Attribute specification for BUFGCE_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFGCE_1 should be placed
-- in architecture after the begin keyword

BUFGCE_1_INSTANCE_NAME : BUFGCE_1
  port map (O => user_O,
           CE => user_CE,
           I => user_I);

Verilog Instantiation Template

BUGCE_1 instance_name (.O (user_O),
             .CE (user_CE),
             .I (user_I));

Commonly Used Constraints

LOC
BUFGDLL

Clock Delay Locked Loop Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>BUFGDLL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

BUFGDLL is a special purpose clock delay locked loop buffer for clock skew management. It is provided as a user convenience for the most frequently used configuration of elements for clock skew management. Internally, it consists of an IBUF driving the CLkin pin of a CLKD LL followed by a BUFG that is driven by the CLK0 pin of the CLKD LL. Because BUFGDLL already contains an input buffer (IBUF), it can only be driven by a top-level port (IPAD).

Any DUTY_CYCLE_CORRECTION attribute on a BUFGDLL applies to the underlying CLKD LL symbol.

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

-- Component Declaration for BUFGDLL should be placed
-- after architecture statement but before begin keyword

component BUFGDLL
    port (O : out STD_ULOGIC;
         I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFGDLL
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
-- Component Instantiation for BUFGDLL should be placed
-- in architecture after the begin keyword

BUFGDLL_INSTANCE_NAME : BUFGDLL
    port map (O => user_O,
                I => user_I);

Verilog Instantiation Template

BUFGDLL instance_name (.O (user_O),
                  .I (user_I));

Commonly Used Constraints

STARTUP_WAIT
BUFGMUX

Global Clock MUX Buffer with Output State 0

Architectures Supported

<table>
<thead>
<tr>
<th>BUFGMUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

BUFGMUX is a multiplexed global clock buffer that can select between two input clocks I0 and I1. When the select input (S) is Low, the signal on I0 is selected for output (O). When the select input (S) is High, the signal on I1 is selected for output.

BUFGMUX and BUFGMUX_1 are distinguished by which state the output assumes when it switches between clocks in response to a change in its select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Using a BUFGMUX element in your design may cause inaccurate simulation if all the following conditions occur: both clock inputs (I0 and I1) are used, GSR is activated during simulation (after simulation time ‘0’), and the secondary clock input (I1) is selected before or while GSR is active. In this case, the primary clock input (I0) is incorrectly selected. This occurs because there is a cross-coupled register pair that ensures the BUFGMUX output does not inadvertently generate a clock edge. When GSR is asserted, these registers initialize to the default state of I0. To select the secondary clock, you must send a clock pulse to both the primary and secondary clock inputs while GSR is inactive.

**Note:** BUFGMUX guarantees that when S is toggled, the state of the output will remain in the inactive state until the next active clock edge (either I0 or I1) occurs.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>I1</td>
</tr>
<tr>
<td>I0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>I1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Usage

This design element is supported for schematics and instantiations but not for inference.
VHDL Instantiation Template

-- Component Declaration for BUFGMUX should be placed
-- after architecture statement but before begin keyword

component BUFGMUX
    port (O : out STD_ULOGIC;
    I0 : in STD_ULOGIC;
    I1 : in STD_ULOGIC;
    S : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFGMUX
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFGMUX should be placed
-- in architecture after the begin keyword

BUFGMUX_INSTANCE_NAME : BUFGMUX
    port map (O => user_O,
    I0 => user_I0,
    I1 => user_I1,
    S => user_S);

Verilog Instantiation Template

BUFGMUX instance_name (.O (user_O),
    .I0 (user_I0),
    .I1 (user_I1),
    .S (user_S));

Commonly Used Constraints

LOC
BUFGMUX_1 is a multiplexed global clock buffer that can select between two input clocks I0 and I1. When the select input (S) is Low, the signal on I0 is selected for output (O). When the select input (S) is High, the signal on I1 is selected for output.

BUFGMUX and BUFGMUX_1 are distinguished by which state the output assumes when it switches between clocks in response to a change in its select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Using a BUFGMUX_1 element in your design may cause inaccurate simulation if all the following conditions occur: both clock inputs (I0 and I1) are used, GSR is activated during simulation (after simulation time ‘0’), and the secondary clock input (I1) is selected before or while GSR is active. In this case, the primary clock input (I0) is incorrectly selected. This occurs because there is a cross-coupled register pair that ensures the BUFGMUX_1 output does not inadvertently generate a clock edge. When GSR is asserted, these registers initialize to the default state of I0. To select the secondary clock, you must send a clock pulse to both the primary and secondary clock inputs while GSR is inactive.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>O</td>
</tr>
<tr>
<td>I1</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>I0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>I1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Usage**

This design element is supported for schematics and instantiations but not for inference.
VHDL Instantiation Template

-- Component Declaration for BUFGMUX_1 should be placed
-- after architecture statement but before begin keyword

component BUFGMUX_1
  port (O : out STD_ULOGIC;
       I0 : in STD_ULOGIC;
       I1 : in STD_ULOGIC;
       S : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFGMUX_1
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes here

-- Component Instantiation for BUFGMUX should be placed
-- in architecture after the begin keyword

BUFGMUX_1_INSTANCE_NAME : BUFGMUX_1
  port map (O => user_O,
            I0 => user_I0,
            I1 => user_I1,
            S => user_S);

Verilog Instantiation Template

BUFGMUX_1 instance_name (.O (user_O),
                        .I0 (user_I0),
                        .I1 (user_I1),
                        .S (user_S));

Commonly Used Constraints

LOC
Primary Global Buffer for Driving Clocks or Longlines (Four per PLD Device)

<table>
<thead>
<tr>
<th>BUFGP</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

BUFGP, a primary global buffer, is used to distribute high fan-out clock or control signals throughout PLD devices.

In Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, BUFGP is equivalent to an IBUFG driving a BUFG.

In XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, BUFGP is treated like BUFG.

A BUFGP provides direct access to Configurable Logic Block (CLB) and Input Output Block (IOB) clock pins and limited access to other CLB inputs. The input to a BUFGP comes only from a dedicated IOB.

Because of its structure, a BUFGP can always access a clock pin directly. However, it can access only one of the F3, G1, C3, or C1 pins, depending on the corner in which the BUFGP is placed. When the required pin cannot be accessed directly from the vertical line, PAR feeds the signal through another CLB and uses general purpose routing to access the load pin.

To use a BUFGP in a schematic, connect the input of the BUFGP element directly to the PAD symbol. Do not use any IBUFs, because the signal comes directly from a dedicated IOB. The output of the BUFGP is then used throughout the schematic. For a negative-edge clock, insert an INV (inverter) element between the output of the BUFGP and the clock input. This inversion is performed inside each CLB or IOB.

A Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, or Spartan-3 BUFGP must be sourced by an external signal.

Usage

This design element is supported for schematic and instantiation. Synthesis tools usually infer a BUFGP on any clock net. If there are more clock nets than BUFGPs, the synthesis tool usually instantiates BUFGPs for the clocks that are most utilized.
VHDL Instantiation Template

-- Component Declaration for BUFGP should be placed
-- after architecture statement but before begin keyword

component BUFGP
  port (O : out STD_ULOGIC;
       I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFGP
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFGP should be placed
-- in architecture after the begin keyword

BUFGP_INSTANCE_NAME : BUFGP
  port map (O => user_O,
            I => user_I);

Verilog Instantiation Template

BUFGP instance_name (.O (user_O),
                       .I (user_I));

Commonly Used Constraints

HBLKNM
Global Set/Reset Input Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

BUFGSR distributes global set/reset signals throughout selected flip-flops of an XC9500/XV/XL, CoolRunner XPLA3, or CoolRunner-II device. Global Set/Reset (GSR) control pins are available on these CPLD devices. Consult device data sheets for availability.

BUFGSR always acts as an input buffer. To use it in a schematic, connect the input of the BUFGSR symbol to an IPAD or an IOPAD representing the GSR signal source. GSR signals generated on-chip must be passed through an OBUF-type buffer before they are connected to BUFGSR.

For global set/reset control, the output of BUFGSR normally connects to the CLR or PRE input of a flip-flop symbol, like FDCP, or any registered symbol with asynchronous clear or preset. The global set/reset control signal may pass through an inverter to perform an active-low set/reset. The output of BUFGSR may also be used as an ordinary input signal to other logic elsewhere in the design. Each BUFGSR can control any number of flip-flops in a design.

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

```
-- Component Declaration for BUFGSR should be placed
-- after architecture statement but before begin keyword

component BUFGSR
    port (O : out STD_ULOGIC;
           I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFGSR
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
```
-- Component Instantiation for BUFGSR should be placed
-- in architecture after the begin keyword

BUFGSR_INSTANCE_NAME : BUFGSR
    port map (O => user_O,
               I => user_I);

Verilog Instantiation Template

BUFGSR instance_name (.O (user_O),
                       .I (user_I));
Global 3-State Input Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>BUFGTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

BUFGTS distributes global output-enable signals throughout the output pad drivers of an XC9500/XV/XL, CoolRunner XPLA3, or CoolRunner-II device. Global Three-State (GTS) control pins are available on these CPLD devices. Consult device data sheets for availability.

BUFGTS always acts as an input buffer. To use it in a schematic, connect the input of the BUFGTS symbol to an IPAD or an IOPAD representing the GTS signal source. GTS signals generated on-chip must be passed through an OBUF-type buffer before they are connected to BUFGTS.

For global 3-state control, the output of BUFGTS normally connects to the E input of a 3-state output buffer symbol, OBUFE. The global 3-state control signal may pass through an inverter or control an OBUFT symbol to perform an active-low output-enable. The same 3-state control signal may even be used both inverted and non-inverted to enable alternate groups of device outputs. The output of BUFGTS may also be used as an ordinary input signal to other logic elsewhere in the design. Each BUFGTS can control any number of output buffers in a design.

Usage

This design element is supported for schematics and instantiations but not for inference.

VHDL Instantiation Template

```
-- Component Declaration for BUFGTS should be placed
-- after architecture statement but before begin keyword

component BUFGTS
    port (O : out STD_ULOGIC;
         I : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFGTS
-- should be placed after architecture declaration but
-- before the begin keyword
```
-- Enter attributes here

-- Component Instantiation for BUFGTS should be placed
-- in architecture after the begin keyword

BUFGTS_INSTANCE_NAME : BUFGTS
    port map (O => user_O,
              I => user_I);

Verilog Instantiation Template

BUFGTS instance_name (.O (user_O),
                        .I (user_I));
Internal 3-State Buffers with Active-Low Enable

Architectures Supported

<table>
<thead>
<tr>
<th>BUFT</th>
<th>BUFT4, BUFT8, BUFT16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XY, XC9500XL *</td>
<td>Macro*</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Not supported for XC9500XL and XC9500XV devices.

BUFT, BUFT4, BUFT8, and BUFT16 are single or multiple 3-state buffers with inputs I, I3 – I0, I7 – I0, and I15 – I0, respectively; outputs O, O3 – O0, O7 – O0, and O15 – O0, respectively; and active-Low output enable (T). When T is Low, data on the inputs of the buffers is transferred to the corresponding outputs. When T is High, the output is high impedance (Z state or off). The outputs of the buffers are connected to horizontal longlines in FPGA architectures.

The outputs of separate BUFT symbols can be tied together to form a bus or a multiplexer. Make sure that only one T is Low at one time.

For XC9500 devices, BUFT output nets assume the High logic level when all connected BUFE/BUFT buffers are disabled.

For Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, when all BUFTs on a net are disabled, the net is High. For correct simulation of this effect, a PULLUP element must be connected to the net. NGDBuild inserts a PULLUP element if one is not connected so that back-annotation simulation reflects the true state of the device.

Usage

These design elements are supported for schematics, instantiations, or inferences.

VHDL Inference Code

```vhdl
architecture Behavioral of buft is
begin
process (I, T)
begin
if (T = '0') then
  O <= I;
else
  O <= 'Z';
end if;
end process;
end Behavioral;
```
Verilog Inference Code

always @(I or T)
begin
    if (!T)
        O <= I;
    else
        O <= 1'bZ;
end

VHDL Instantiation Template

-- Component Declaration for BUFT should be placed
-- after architecture statement but before begin keyword

component BUFT
    port (O : out STD_ULOGIC;
            I : in STD_ULOGIC;
            T : in STD_ULOGIC);
end component;

-- Component Attribute specification for BUFT
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for BUFT should be placed
-- in architecture after the begin keyword

BUFT_INSTANCE_NAME : BUFT
    port map (O => user_O,
            I => user_I,
            T => user_T);

Verilog Instantiation Template

BUFT instance_name (.O (user_O),
                    .I (user_I),
                    .T (user_T));

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
LOC
U_SET
CAPTURE_SPARTAN2

Spartan-II Register State Capture for Bitstream Readback

Architectures Supported

<table>
<thead>
<tr>
<th>CAPTURE_SPARTAN2</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

CAPTURE_SPARTAN2 provides user control over when to capture register (flip-flop and latch) information for readback. Spartan-II and Spartan-IIE devices provide the readback function through dedicated configuration port instructions. The CAPTURE_SPARTAN2 symbol is optional. Without it readback is still performed, but the asynchronous capture function it provides for register states is not available.

Note: Spartan-II and Spartan-IIE devices only allow for capturing register (flip-flop and latch) states. Although LUT RAM, SRL, and block RAM states are read back, they cannot be captured.

An asserted High CAP signal indicates that the registers in the device are to be captured at the next Low-to-High clock transition. By default, data is captured after every trigger (transition on CLK while CAP is asserted). To limit the readback operation to a single data capture, add the ONESHOT attribute to CAPTURE_SPARTAN2. See the Constraints Guide for information on the ONESHOT attribute.

Note: For details on the Spartan-II and Spartan-IIE readback functions, see The Programmable Logic Data Book.

Usage

This design element is supported for schematics and instantiation but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CAPTURE_SPARTAN2 should be placed after architecture statement but before begin keyword
component CAPTURE_SPARTAN2
    port (CAP : in STD_ULOGIC;
         CLK : in STD_ULOGIC);
end component;

-- Component Attribute specification for CAPTURE_SPARTAN2 should be placed after architecture declaration but
```
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for CAPTURE_SPARTAN2 should be
-- placed in architecture after the begin keyword

CAPTURE_SPARTAN2_INSTANCE_NAME : CAPTURE_SPARTAN2
    port map (CAP => user_CAP,
                CLK => user_CLK);

Verilog Instantiation Template

CAPTURE_SPARTAN2 instance_name (.CAP (user_CAP),
                                .CLK (user_CLK));

Commonly Used Constraints

ONESHOT
CAPTURE_SPARTAN3

Spartan-3 Register State Capture for Bitstream Readback

Architectures Supported

<table>
<thead>
<tr>
<th>CAPTURE_SPARTAN3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

CAPTURE_SPARTAN3 provides user control over when to capture register (flip-flop and latch) information for readback. Spartan-3 devices provide the readback function through dedicated configuration port instructions.

The CAPTURE_SPARTAN3 symbol is optional. Without it readback is still performed, but the asynchronous capture function it provides for register states is not available.

Spartan-3 allows for capturing register (flip-flop and latch) states only. Although LUT RAM, SRL, and block RAM states are read back, they cannot be captured. An asserted high CAP signal indicates that the registers in the device are to be captured at the next Low-to-High clock transition.

By default, data is captured after every trigger (transition on CLK while CAP is asserted). To limit the readback operation to a single data capture, add the ONESHOT attribute to CAPTURE_SPARTAN3. See the Constraints Guide for information on the ONESHOT attribute.

Usage

This design element is instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CAPTURE_SPARTAN3 should be
-- placed after architecture statement but before
-- begin keyword

component CAPTURE_SPARTAN3
    port (CAP : in STD_ULOGIC;
    CLK : in STD_ULOGIC);
end component;

-- Component Attribute specification for CAPTURE_SPARTAN3
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
```
-- Component Instantiation for CAPTURE_SPARTAN3 should be
-- placed in architecture after the begin keyword

CAPTURE_SPARTAN3_INSTANCE_NAME : CAPTURE_SPARTAN3
    port map (CAP => user_CAP,
             CLK => user_CLK);

Verilog Instantiation Template

CAPTURE_SPARTAN3 instance_name (.CAP (user_CAP),
                                 .CLK (user_CLK));

Commonly Used Constraints

ONESHOT
Virtex Register State Capture for Bitstream Readback

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

CAPTURE_VIRTEX provides user control over when to capture register (flip-flop and latch) information for readback. Virtex and Virtex-E devices provide the readback function through dedicated configuration port instructions.

The CAPTURE_VIRTEX symbol is optional. Without it readback is still performed, but the asynchronous capture function it provides for register states is not available.

**Note:** Virtex and Virtex-E allow for capturing register (flip-flop and latch) states only. Although LUT RAM, SRL, and block RAM states are read back, they cannot be captured.

An asserted High CAP signal indicates that the registers in the device are to be captured at the next Low-to-High clock transition. By default, data is captured after every trigger (transition on CLK while CAP is asserted). To limit the readback operation to a single data capture, add the ONESHOT attribute to CAPTURE_VIRTEX. See the *Constraints Guide* for information on the ONESHOT attribute.


**Usage**

This design element is supported for schematics and instantiation but not for inference.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for CAPTURE_VIRTEX should be
-- placed after architecture statement but before
-- begin keyword

component CAPTURE_VIRTEX
  port (CAP : in STD_ULOGIC;
       CLK : in STD_ULOGIC);
end component;

-- Component Attribute specification for CAPTURE_VIRTEX
-- should be placed after architecture declaration but
-- before the begin keyword
```
-- Enter attributes here

-- Component Instantiation for CAPTURE_VIRTEX should be placed in architecture after the begin keyword

CAPTURE_VIRTEX_INSTANCE_NAME : CAPTURE_VIRTEX
  port map (CAP => user_CAP,
          CLK => user_CLK);

Verilog Instantiation Template

CAPTURE_VIRTEX instance_name (.CAP (user_CAP),
                              .CLK (user_CLK));

Commonly Used Constraints

ONESHOT
CAPTURE_VIRTEX2

Virtex-II Register State Capture for Bitstream Readback

Architectures Supported

<table>
<thead>
<tr>
<th>CAPTURE_VIRTEX2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

CAPTURE_VIRTEX2 provides user control over when to capture register (flip-flop and latch) information for readback. Virtex-II, Virtex-II Pro, and Virtex-II Pro X devices provide the readback function through dedicated configuration port instructions.

The CAPTURE_VIRTEX2 symbol is optional. Without it readback is still performed, but the asynchronous capture function it provides for register states is not available.

Virtex-II, Virtex-II Pro, and Virtex-II Pro X allow for capturing register (flip-flop and latch) states only. Although LUT RAM, SRL, and block RAM states are read back, they cannot be captured.

An asserted high CAP signal indicates that the registers in the device are to be captured at the next Low-to-High clock transition. By default, data is captured after every trigger (transition on CLK while CAP is asserted). To limit the readback operation to a single data capture, add the ONESHOT attribute to CAPTURE_VIRTEX2. See the Constraints Guide for information on the ONESHOT attribute.

The GRDBK (internal capture signal) is asserted at the rising edge of the Capture clock when the Capture signal is high. The capture memory cell is continuously updated if the value of the FF is continuously changing. The final value of the capture memory cell occurs at the rising edge of the Capture clock when the Capture signal is low.

For details on the Virtex-II, Virtex-II Pro, and Virtex-II Pro X readback functions, see The Programmable Logic Data Book.

Usage

This design element is supported for schematics and instantiation but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CAPTURE_VIRTEX2 should be
-- placed after architecture statement but before
-- begin keyword
```
component CAPTURE_VIRTEX2
    port (CAP : in STD_ULOGIC;
         CLK : in STD_ULOGIC);
end component;

-- Component Attribute specification for CAPTURE_VIRTEX2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for CAPTURE_VIRTEX2 should be
-- placed in architecture after the begin keyword

CAPTURE_VIRTEX2_INSTANCE_NAME : CAPTURE_VIRTEX2
    port map (CAP => user_CAP,
              CLK => user_CLK);

Verilog Instantiation Template

CAPTURE_VIRTEX2 instance_name (.
                                 CAP (user_CAP),
                                 .CLK (user_CLK));

Commonly Used Constraints

None
CB2CE, CB4CE, CB8CE, CB16CE

2-, 4-, 8-, 16-Bit Cascadable Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CB2CE, CB4CE, CB8CE, CB16CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CB2CE, CB4CE, CB8CE, and CB16CE are, respectively, 2-, 4-, 8-, and 16-bit (stage), asynchronous, clearable, cascadable binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when both TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEx, or STARTUP_VIRTEx2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 1 for CB2CE; z = 3 for CB4CE; z = 7 for CB8CE; z = 15 for CB16CE

TC = Qz•Q(z-1)•Q(z-2)•...•Q0

CEO = TC•CE
CB8CE Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E,
Virtex-II, Virtex-II Pro, Virtex-II Pro X

CB2CE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
CB8CE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of cb2ce is

  constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
    (others => '1');

begin

  process(C, CLR)
  begin
    if (CLR='1') then
      Q <= (others => '0');
    elsif (C'event and C='1') then
      Q <= (others => '0');
    else
      Q <= (others => '0');
    end if;

  end process;

end Behavioral;
```
if (CE='1') then
  Q <= Q+1;
end if;
end if;
end process;

process (Q)
begin
  if (Q = TERMINAL_COUNT) then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO<=TC and CE;

end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    Q <= Q + 1;
end

always @ (Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end

always @ (CE or TC)
begin
  CEO <= TC && CE;
end
CB2CLE, CB4CLE, CB8CLE, CB16CLE

2-, 4-, 8-, 16-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CB2CLE, CB4CLE, CB8CLE, CB16CLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CB2CLE, CB4CLE, CB8CLE, and CB16CLE are, respectively, 2-, 4-, 8-, and 16-bit (stage) synchronously loadable, asynchronously clearable, cascadable binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock transition, independent of the state of clock enable (CE). The Q outputs increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CB2CLE; z = 3 for CB4CLE; z = 7 for CB8CLE; z = 15 for CB16CLE

TC = Qz•Q(z-1)•Q(z-2)•…•Q0

CEO = TC•CE

CB8CLE Implementation Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
CB2CLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

CB8CLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of cb2cle is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');

begin

process(C, CLR)
begin
  if (CLR='1') then
    Q <= (others => '0');
  elsif (C'event and C='1') then
    if (L = '1') then
      Q <= D;
    elsif (CE='1') then
      Q <= COUNT+1;
    end if;
  end if;
end process;

process(Q)
begin
  if (Q = TERMINAL_COUNT) then
    TC <='1';
  else
    TC <='0';
  end if;
end process;

TC <= TC_INT;
CEO <= TC and CE;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    Q <= Q + 1;
end

always @(Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
```
end

always @ (TC or CE)
begin
  CEO <= TC & CE;
end
CB2CLED, CB4CLED, CB8CLED, CB16CLED

2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CB2CLED, CB4CLED, CB8CLED, CB16CLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CB2CLED, CB4CLED, CB8CLED, and CB16CLED are, respectively, 2-, 4-, 8- and 16-bit (stage), synchronously loadable, asynchronously clearable, cascadable, bidirectional binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The Q outputs decrement when CE is High and UP is Low during the Low-to-High clock transition. The Q outputs increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the CEO output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage.

When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, see “CB2X1, CB4X1, CB8X1, CB16X1” for high-performance cascadable, bidirectional counters.

The counter is asynchronously cleared, output Low, when power is applied. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted with an inverter in front of the GSR input of STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2.
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TC = (Qz•Q(z-1)•Q(z-2)•…•Q0•UP) + (Qz•Q(z-1)•Q(z-2)•…•Q0•UP)

CEO = TC•CE
CB8CLED Implementation Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
CB4CLED Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of cb2cled is

constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) :=
(others => '1');
constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) :=
(others => '0');

begin

process(C, CLR)
begin
if (CLR='1') then
  Q <= (others => '0');
elsif C'event and C='1' then
  if (L = '1') then
    Q <= D;
  elsif (CE='1') then
    if (UP='1') then
      Q <= Q+1;
    elsif (UP='0') then
      Q <= Q-1;
    end if;
  end if;
end if;
end if;
end process;

process(Q, UP)
begin
  if (((Q = TERMINAL_COUNT_UP) and (UP = '1')) or
      ((Q = TERMINAL_COUNT_DOWN) and (UP = '0'))) then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO<=TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    begin
      if (UP)
        Q <= Q + 1;
      else if (!UP)
        Q <= Q - 1;
      end
    end
end
always @(Q or UP)
beg

if((Q == TERMINAL_COUNT_UP && UP) || (Q == TERMINAL_COUNT_DOWN && !UP))
TC <= 1;
else
TC <= 0;
end

always @(TC or CE)
begin
CEO <= TC & CE;
end
CB2RE, CB4RE, CB8RE, CB16RE

2-, 4-, 8-, 16-Bit Cascadable Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>CB2RE, CB4RE, CB8RE, CB16RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CB2RE, CB4RE, CB8RE, and CB16RE are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronous, resettable, cascadable binary counters. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero during the Low-to-High clock transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when both Q outputs are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R )</td>
<td>( CE )</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>Inc</td>
</tr>
<tr>
<td>Qző – Q0</td>
<td>TC</td>
</tr>
</tbody>
</table>

z = 1 for CB2RE; z = 3 for CB4RE; z = 7 for CB8RE; z = 15 for CB16RE

TC = Qz • Q(z-1) • Q(z-2) • ... • Q0

CEO = TC • CE

CB2RE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of cb2re is
constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');

begin

  process(C)
  begin
    if (C'event and C='1') then
      if (R='1') then
        Q <= (others => '0');
      elsif (CE='1') then
        Q <= Q+1;
      end if;
    end if;
  end process;

  process(Q)
  begin
    if (Q = TERMINAL_COUNT) then
      TC <= '1';
    else
      TC <= '0';
    end if;
  end process;

  CEO <= TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (CE)
    Q <= Q + 1;
end

always @ (Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end

always @ (TC or CE)
begin
  CEO <= TC & CE;
end
CB2RLE, CB4RLE, CB8RLE, CB16RLE

2-, 4-, 8-, 16-Bit Loadable Cascadable Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
<th>XC9500, XC9500XV, XC9500XL</th>
<th>CoolRunner XPLA3</th>
<th>CoolRunner-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB2RLE, CB4RLE, CB8RLE, CB16RLE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CB2RLE, CB4RLE, CB8RLE, and CB16RLE are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronous, loadable, resettable, cascadable binary counters. The synchronous reset (R) is the highest priority input. The synchronous R, when High, overrides all other inputs and resets the Q outputs, terminal count (TC), and clock enable out (CEO) outputs to Low on the Low-to-High clock (C) transition.

The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of CE. The Q outputs increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High. The CEO output is High when all Q outputs and CE are High to allow direct cascading of counters.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and by connecting the C, L, and R inputs in parallel. The maximum length of the counter is determined by the accumulated CE-to-CEO propagation delays versus the clock period. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
<td>CE</td>
<td>C</td>
<td>Dz – D0</td>
<td>Qz – Q0</td>
<td>TC</td>
<td>CEO</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>↑</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>↑</td>
<td>Dz</td>
<td>Dz</td>
<td>TC</td>
<td>CEO</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>No Chg</td>
<td>No Chg</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
CB2RLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

CB8RLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of cb2rle is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');
begin
  process(C)
  begin
    if (C'event and C='1') then
      if (R='1') then
        Q <= (others => '0');
      elsif (L='1') then
        Q <= D;
      elsif (CE='1') then
        Q <= Q+1;
      end if;
    end if;
  end process;

  process(Q)
  begin
    if (Q = TERMINAL_COUNT) then
      TC <='1';
    else
      TC <='0';
    end if;
  end process;

  CEO<=TC and CE;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
  if (R)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    Q <= Q + 1;
end

always @(Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end

always @(TC or CE)
begin
  CEO <= TC & CE;
end
CB2X1, CB4X1, CB8X1, CB16X1

2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>CB2X1, CB4X1</th>
<th>CB8X1, CB16X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CB2X1, CB4X1, CB8X1, and CB16X1 are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronously loadable, asynchronously clearable, bidirectional binary counters. These counters have separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; data outputs (Q) go to logic level zero, terminal count outputs TCU and TCD go to zero and one, respectively, clock enable outputs CEOU and CEOD go to Low and High, respectively, independent of clock transitions. The data on the D inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The Q outputs increment when CEU is High, provided CLR and L are Low, during the Low-to-High clock transition. The Q outputs decrement when CED is High, provided CLR and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOD output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOD outputs of each counter are
connected directly to the CEU and CED inputs, respectively, of the next stage. The
clock, L, and CLR inputs are connected in parallel.

The maximum clocking frequency of these counter components is unaffected by the
number of cascaded stages for all counting and loading functions. The TCU terminal
count output is High when all Q outputs are High, regardless of CEU. The TCD
output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND
wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the
down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable
AND gates within the component. This results in zero propagation from the CEU and
CED inputs and from the Q outputs, provided all connections from each such output
remain on-chip. Otherwise, a macrocell buffer delay is introduced.

The counter is initialized to zero (TCU Low and TCD High) when power is applied.
The power-on condition can be simulated by applying a High-level pulse on the
PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CB2X1; z = 3 for CB4X1; z = 7 for CB8X1; z = 15 for CB16X1

TCU = Qz • Q(z-1) • Q(z-2) • ... • Q0

TCD = Qz • Q(z-1) • Q(z-2) • ... • Q0

CEOU = TCU • CEU

CEOD = TCD • CED
CB4X1 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of cb2x1 is

    signal CEU_INT : std_logic;
    signal CED_INT : std_logic;

    constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) := (others => '1');
    constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) := (others => '0');

begin

    process(C, CLR)
    begin
        if (CLR='1') then
            Q <= (others => '0');
            CEU_INT <= '0';
            CED_INT <= '1';
        elsif ((C'event) and (C='1')) then
            if (L = '1') then
                Q <= D;
                CEU_INT <= CEU;
                CED_INT <= CED;
            elsif (CEU='1') then
                Q <= Q+1;
                CEU_INT <= '1';
                CED_INT <= '0';
            elsif (CED='1') then
                Q <= Q-1;
                CEU_INT <= '0';
                CED_INT <= '1';
            end if;
        end if;
    end if;
end process;

    process (Q, CEU_INT, CED_INT)
    begin
        if ((Q = TERMINAL_COUNT_UP) and (CEU_INT = '1')) then
            TCU <= '1';
            TCD <= '0';
        elsif ((Q = TERMINAL_COUNT_DOWN) and (CED_INT = '1')) then
            TCU <= '0';
            TCD <= '1';
        else
            TCU <= '0';
            TCD <= '0';
        end if;
    end process;

    CEOU <= TCU and CEU;
    CEOD <= TCD and CED;
end Behavioral;
Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
    if (CLR)
        begin
            Q <= 0;
            CEU_INT <= 1'b0;
            CED_INT <= 1'b1;
        end
    else if (L)
        begin
            Q <= D;
            CEU_INT <= CEU;
            CED_INT <= CED;
        end
    else if (CEU)
        begin
            Q <= Q + 1;
            CEU_INT <= 1;
            CED_INT <= 0;
        end
    else if (CED)
        begin
            Q <= Q - 1;
            CEU_INT <= 1'b0;
            CED_INT <= 1'b1;
        end
end

always @ (Q or CEU_INT or CED_INT)
begin
    if (Q == TERMINAL_COUNT_UP && CEU_INT)
        begin
            TCU_INT <= 1;
            TCD_INT <= 0;
        end
    else if ((Q == TERMINAL_COUNT_DOWN) && CED_INT)
        begin
            TCU_INT <= 0;
            TCD_INT <= 1;
        end
    else
        begin
            TCU_INT <= 0;
            TCD_INT <= 0;
        end
end

always @ (TCU_INT or CEU or TCD_INT or CED)
begin
    CEOU <= TCU_INT && CEU;
    CEOD <= TCD_INT && CED;
    TCU <= TCU_INT;
    TCD <= TCD_INT;
end
CB2X2, CB4X2, CB8X2, CB16X2

2-, 4-, 8-, and 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>CB2X2</th>
<th>CB4X2, CB8X2, CB16X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CB2X2, CB4X2, CB8X2, and CB16X2 are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronous, loadable, resettable, bidirectional binary counters. These counters have separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in CPLD architectures.

The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored; the data outputs (Q) go to logic level zero, terminal count outputs TCU and TCD go to zero and one, respectively, and clock enable outputs CEOU and CEDO go to Low and High, respectively, on the Low-to-High clock (C) transition. The data on the D inputs loads into the counter on the Low-to-High clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

All Q outputs increment when CEOU is High, provided R and L are Low during the Low-to-High clock transition. All Q outputs decrement when CED is High, provided R and L are Low. The counter ignores clock transitions when CEOU and CEDO are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEDO outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEDO output is High when all Q outputs and CED are High. To cascade counters, the CEOU and CEDO outputs of each counter are, respectively, connected directly to the CEU and CED inputs of the next stage. The C, L, and R inputs are connected in parallel.
The maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable AND gates within the component. This results in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced.

The counter is initialized to zero (TCU Low and TCD High) when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CB2X2; z = 3 for CB4X2; z = 7 for CB8X2; z = 15 for CB16X2

TCU = Qz • Q(z-1) • Q(z-2) • ... • Q0
TCD = Qz • Q(z-1) • Q(z-2) • ... • Q0
CEOU = TCU • CEU
CEOD = TCD • CED
CB4X2 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```
architecture Behavioral of cb2x2 is

    signal CEU_INT : std_logic;
    signal CED_INT : std_logic;
    constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) := (others => '1');
    constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) := (others => '0');

    begin

        process(C)
        begin
            if ((C'event) and (C='1'))then
                if (R='1') then
                    Q <= (others => '0');
                    CEU_INT <= '0';
                    CED_INT <= '1';
                elsif (L = '1') then
                    Q <= D;
                    CEU_INT <= CEU;
                    CED_INT <= CED;
                elsif (CEU='1') then
                    Q <= Q+1;
                    CEU_INT <= '1';
                    CED_INT <= '0';
                elsif (CED='1') then
                    Q <= Q-1;
                    CEU_INT <= '0';
                    CED_INT <= '1';
                end if;
            end if;
        end process;

        process(Q, CEU_INT, CED_INT)
        begin
            if ((Q = TERMINAL_COUNT_UP) and (CEU_INT = '1')) then
                TCU <= '1';
                TCD <= '0';
            elsif ((Q = TERMINAL_COUNT_DOWN) and (CED_INT = '1')) then
                TCU <= '0';
                TCD <= '1';
            else
                TCU <= '0';
                TCD <= '0';
            end if;
        end process;

        CEOU<=TCU and CEU;
        CEOD<=TCD and CED;
```
end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
if (R)
begin
Q <= 0;
CEU_INT <= 1'b0;
CED_INT <= 1'b1;
end
else if (L)
begin
Q <= D;
CEU_INT <= CEU;
CED_INT <= CED;
end
else if (CEU)
begin
Q <= Q + 1;
CEU_INT <= 1;
CED_INT <= 0;
end
else if (CED)
begin
Q <= Q - 1;
CEU_INT <= 1'b0;
CED_INT <= 1'b1;
end
end

always @ (Q or CEU_INT or CED_INT)
begin
if (Q == TERMINAL_COUNT_UP && CEU_INT)
begin
TCU_INT <= 1;
TCD_INT <= 0;
end
else if ((Q == TERMINAL_COUNT_DOWN) && CED_INT)
begin
TCU_INT <= 0;
TCD_INT <= 1;
end
else
begin
TCU_INT <= 0;
TCD_INT <= 0;
end
end

always @(TCU_INT or CEU or TCD_INT or CED)
begin
CEOU <= TCU_INT && CEU;
CEOD <= TCD_INT && CED;
TCU <= TCU_INT;
TCD <= TCD_INT;
end
CBD2CE, CBD4CE, CBD8CE, CBD16CE

2-, 4-, 8-,16-Bit Cascadable Dual Edge Triggered Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CBD2CE, CBD4CE, CBD8CE, CBD16CE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CBD2CE, CBD4CE, CBD8CE, and CBD16CE are, respectively, 2-, 4-, 8-, and 16-bit (stage), asynchronous, clearable, cascadable dual edge triggered binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High and High-to-Low clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, outputs Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.
CBD2CE Implementation CoolRunner-II

CBD8CE Implementation CoolRunner-II
Usage

For HDL, these design elements are supported for inference but not instantiation.

VHDL Inference Code

```vhdl
architecture Behavioral of cbd2ce is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');

begin

process(C, CLR)
begin
  if (CLR='1') then
    Q <= (others => '0');
  elsif C'event then
    if (CE='1') then
      Q <= Q+1;
    end if;
  end if;
end process;

process (Q)
begin
  if (Q = TERMINAL_COUNT) then
    TC <='1';
  else
    TC <='0';
  end if;
end process;

CEO<=TC and CE;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    Q <= Q + 1;
end

always @ (Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end

always @ (TC or CE)
begin
  CEO <= TC & CE;
end
```
CBD2CLE, CBD4CLE, CBD8CLE, CBD16CLE

2-, 4-, 8-, 16-Bit Loadable Cascadable Dual Edge Triggered Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

| CBD2CLE, CBD4CLE, CBD8CLE, CBD16CLE | Spartan-II, Spartan-IIIE | No |
| Spartan-3 | No |
| Virtex, Virtex-E | No |
| Virtex-II, Virtex-II Pro, Virtex-II Pro X | No |
| XC9500, XC9500XV, XC9500XL | No |
| CoolRunner XPLA3 | No |
| CoolRunner-II | Macro |

CBD2CLE, CBD4CLE, CBD8CLE, and CBD16CLE are, respectively, 2-, 4-, 8-, and 16-bit (stage) synchronously loadable, asynchronously clearable, cascadable dual edge triggered binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock transition, independent of the state of clock enable (CE). The Q outputs increment when CE is High during the Low-to-High clock transition, independent of the state of clock enable (CE). The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
CBD2CLE Implementation CoolRunner-II

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( z = 1 \) for CBD2CLE; \( z = 3 \) for CBD4CLE; \( z = 7 \) for CBD8CLE; \( z = 15 \) for CBD16CLE

\( TC = Qz \cdot Q(z-1) \cdot Q(z-2) \ldots \cdot Q0 \)

CEO = TC \cdot CE
CBD8CLE Implementation CoolRunner-II

Usage

For HDL, these design elements are supported for inference but not instantiation.

VHDL Inference Code

```vhdl
architecture Behavioral of cbd2cle is

    constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
        (others => '1');

begin

    process(C, CLR)
    begin
        if (CLR='1') then
```
Q <= (others => '0');
elif C'event then
  if (L = '1') then
    Q <= D;
elif (CE='1') then
    Q <= Q+1;
  end if;
end if;
end process;

process(Q)
begin
  if (Q = TERMINAL_COUNT) then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO<=TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    Q <= Q + 1;
end

always @ (Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end

always @ (TC or CE)
begin
  CEO <= TC & CE;
end
CBD2CLED, CBD4CLED, CBD8CLED, CBD16CLED

2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CBD2CLED</th>
<th>CBD4CLED</th>
<th>CBD8CLED</th>
<th>CBD16CLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CBD2CLED, CBD4CLED, CBD8CLED, and CBD16CLED are, respectively, 2-, 4-, 8- and 16-bit (stage), synchronously loadable, asynchronously clearable, cascadable, bidirectional dual edge triggered binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The Q outputs decrement when CE is High and UP is Low during the Low-to-High and High-to-Low clock transition. The Q outputs increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the CEO output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage.

When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not. For CoolRunner-II, see “CB2X1, CB4X1, CB8X1, CB16X1” for high-performance cascadable, bidirectional counters.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CBD2CLED; z = 3 for CBD4CLED; z = 7 for CBD8CLED; z = 15 for CBD16CLED
TC = (Qz•Q(z-1)•Q(z-2)•...•Q0•UP) + (Qz•Q(z-1)•Q(z-2)•...•Q0•UP) + (Qz•Q(z-1)•Q(z-2)•...•Q0•UP)
CEO = TC•CE
CBD4CLED Implementation CoolRunner-II
Usage

For HDL, these design elements are supported for inference but not instantiation.

VHDL Inference Code

```vhdl
architecture Behavioral of cbd2cled is

constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');
constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '0');

begin

process(C, CLR)
begin
  if (CLR='1') then
    Q <= (others => '0');
  elsif (C'event) then
    if (L = '1') then
      Q <= D;
    elsif (CE='1') then
      if (UP='1') then
        Q <= Q+1;
      elsif (UP='0') then
        Q <= Q-1;
      end if;
    end if;
  end if;
end process;

process(Q, UP)
begin
  if (((Q = TERMINAL_COUNT_UP) and (UP = '1')) or
      ((Q = TERMINAL_COUNT_DOWN) and (UP = '0'))) then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO <= TC and CE;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    begin
      if (UP)
        Q <= Q + 1;
```
else if (!UP)
    Q <= Q - 1;
end
end

always @ (Q or UP)
begin
    if ((Q == TERMINAL_COUNT_UP && UP) || (Q == TERMINAL_COUNT_DOWN && !UP))
        TC <= 1;
    else
        TC <= 0;
end

always @ (TC or CE)
begin
    CEO <= TC & CE;
end
CBD2RE, CBD4RE, CBD8RE, CBD16RE

2-, 4-, 8-, 16-Bit Cascadable Dual Edge Triggered Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>CBD2RE, CBD4RE, CBD8RE, CBD16RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CBD2RE, CBD4RE, CBD8RE, and CBD16RE are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronous, resettable, cascadable dual edge triggered binary counters. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero during the Low-to-High or High-to-Low clock transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High and High-to-Low clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when both Q outputs are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Qz – Q0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X 1</td>
</tr>
<tr>
<td>0</td>
<td>0 1</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>R</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 1 for CBD2RE; z = 3 for CBD4RE; z = 7 for CBD8RE; z = 15 for CBD16RE

TC = Qz•Q(z-1)•Q(z-2)•...•Q0

CEO = TC•CE

CBD2RE Implementation CoolRunner-II
CBD8RE Implementation CoolRunner-II

Usage

For HDL, these design elements are supported for inference but not instantiation.
VHDL Inference Code

architecture Behavioral of cbd2re is

    constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
    (others => '1');

begin
    process(C, R)
    begin
        if (C'event) then
            if (R='1') then
                Q <= (others => '0');
            elsif (CE='1') then
                Q <= Q+1;
            end if;
        end if;
    end process;

    process(Q)
    begin
        if (Q = TERMINAL_COUNT) then
            TC <= '1';
        else
            TC <= '0';
        end if;
    end process;

    CEO <= TC and CE;
end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C)
begin
    if (R)
        Q <= 0;
    else if (CE)
        Q <= Q + 1;
end

always @(Q)
begin
    if (Q == TERMINAL_COUNT)
        TC <= 1;
    else
        TC <= 0;
end

always @(TC or CE)
begin
    CEO <= TC & CE;
end
CBD2RLE, CBD4RLE, CBD8RLE, CBD16RLE

2-, 4-, 8-, 16-Bit Loadable Cascadable Dual Edge Triggered Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
<th>XC9500, XC9500XV, XC9500XL</th>
<th>CoolRunner XPLA3</th>
<th>CoolRunner-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBD2RLE, CBD4RLE, CBD8RLE, CBD16RLE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CBD2RLE, CBD4RLE, CBD8RLE, and CBD16RLE are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronous, loadable, resettable, cascadable dual edge triggered binary counters. The synchronous reset (R) is the highest priority input. The synchronous R, when High, overrides all other inputs and resets the Q outputs, terminal count (TC), and clock enable out (CEO) outputs to Low on the Low-to-High or High-to-Low clock (C) transition.

The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High and High-to-Low clock (C) transition, independent of the state of CE. The Q outputs increment when CE is High during the Low-to-High and High-to-Low clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High. The CEO output is High when all Q outputs and CE are High to allow direct cascading of counters.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and by connecting the C, L, and R inputs in parallel. The maximum length of the counter is determined by the accumulated CE-to-CEO propagation delays versus the clock period. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CBD2RLE; z = 3 for CBD4RLE; z = 7 for CBD8RLE; z = 15 for CBD16RLE
TC = Qz•Q(z-1)•Q(z-2)•...•Q0
CEO = TC•CE

CBD2RLE Implementation CoolRunner-II
CBD8RLE Implementation CoolRunner-II

Usage
For HDL, these design elements are supported for inference but not instantiation.

VHDL Inference Code

```vhdl
architecture Behavioral of cbd2rle is

  constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
    (others => '1');

begin

  process(C, R)
  begin
    if (C'event) then
      if (R='1') then
        Q <= (others => '0');
      elsif (L='1') then
        Q <= D;
      elsif (CE='1') then
        Q <= Q+1;
      end if;
    end if;
  end process;

  process(Q)
  begin
    process
  end process;

end Behavioral;
```
if (Q == TERMINAL_COUNT) then
   TC <= '1';
else
   TC <= '0';
end if;
end process;

CEO <= TC and CE;

end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
   if (R)
      Q <= 0;
   else if (L)
      Q <= D;
   else if (CE)
      Q <= Q + 1;
   end

always @ (Q)
begin
   if (Q == TERMINAL_COUNT)
      TC <= 1;
   else
      TC <= 0;
   end

always @ (TC or CE)
begin
   CEO <= TC & CE;
end
CBD2X1, CBD4X1, CBD8X1, CBD16X1

2-, 4-, 8-, 16-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CBD2X1, CBD4X1, CBD8X1, CBD16X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CBD2X1, CBD4X1, CBD8X1, and CBD16X1 are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronously loadable, asynchronously clearable, bidirectional dual edge triggered binary counters. These counters have separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the CoolRunner-II architecture.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; data outputs (Q) go to logic level zero, terminal count outputs TCU and TCD go to zero and one, respectively, clock enable outputs CEOU and CEOD go to Low and High, respectively, independent of clock transitions. The data on the D inputs loads into the counter on the Low-to-High and High-to-Low clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

The Q outputs increment when CEU is High, provided CLR and L are Low, during the Low-to-High and High-to-Low clock transition. The Q outputs decrement when CED is High, provided CLR and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CEOU output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CEOE outputs of each counter are connected directly to the CEU and CED inputs, respectively, of the next stage. The clock, L, and CLR inputs are connected in parallel.

In CoolRunner-II, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable
AND gates within the component. This results in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced.

The counter is initialized to zero (TCU Low and TCD High) when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CBD2X1; z = 3 for CBD4X1; z = 7 for CBD8X1; z = 15 for CBD16X1
TCU = Qz•Q(z-1)•Q(z-2)•...•Q0
TCD = Qz•Q(z-1)•Q(z-2)•...•Q0
CEOU = TCU•CEU
CEOD = TCD•CED

CBD4X1 Implementation CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of cbd2x1 is

    signal CEU_INT : std_logic;
    signal CED_INT : std_logic;

    constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) :=
        (others => '1');
    constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) :=
        (others => '0');

begin

    process(C, CLR)
    begin
        if (CLR='1') then
            Q <= (others => '0');
            CEU_INT <= '0';
            CED_INT <= '1';
        elsif C'event then
            if (L = '1') then
                Q <= D;
                CEU_INT <= CEU;
                CED_INT <= CED;
            elsif (CEU='1') then
                Q <= Q+1;
                CEU_INT <= '1';
                CED_INT <= '0';
            elsif (CED='1') then
                Q <= Q-1;
                CEU_INT <= '0';
                CED_INT <= '1';
            end if;
        end if;
    end if;
end process;

    process(Q, CEU_INT, CED_INT)
    begin
        if ((Q = TERMINAL_COUNT_UP) and (CEU_INT = '1')) then
            TCU_INT <= '1';
            TCD_INT <= '0';
        elsif ((Q = TERMINAL_COUNT_DOWN) and (CED_INT = '1')) then
            TCU <= '0';
            TCD <= '1';
        else
            TCU <= '0';
            TCD <= '0';
        end if;
    end process;

    CEOU <= TCU and CEU;
    CEOD <= TCD and CED;

end Behavioral;
Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
begin
if (CLR)
begin
Q <= 0;
CEU_INT <= 1'b0;
CED_INT <= 1'b1;
end
else if (L)
begin
Q <= D;
CEU_INT <= CEU;
CED_INT <= CED;
end
else if (CEU)
begin
Q <= Q + 1;
CEU_INT <= 1;
CED_INT <= 0;
end
else if (CED)
begin
Q <= Q - 1;
CEU_INT <= 1'b0;
CED_INT <= 1'b1;
end
end

always @ (Q or CEU_INT or CED_INT)
begin
if (Q == TERMINAL_COUNT_UP && CEU_INT)
begin
TCU_INT <= 1;
TCD_INT <= 0;
end
else if ((Q == TERMINAL_COUNT_DOWN) && CED_INT)
begin
TCU_INT <= 0;
TCD_INT <= 1;
end
else
begin
TCU_INT <= 0;
TCD_INT <= 0;
end
end

always @ (TCU_INT or CEU or TCD_INT or CED)
begin
CEOU <= TCU_INT && CEU;
CEED <= TCD_INT && CED;
TCU <= TCU_INT;
TCD <= TCD_INT;
end
CBD2X2, CBD4X2, CBD8X2, CBD16X2

2-, 4-, 8-, and 16-Bit Loadable Cascadable Bidirectional Dual Edge Triggered Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CBD2X2, CBD4X2, CBD8X2, and CBD16X2 are, respectively, 2-, 4-, 8-, and 16-bit (stage), synchronous, loadable, resettable, bidirectional dual edge triggered binary counters. These counters have separate count-enable inputs and synchronous terminal-count outputs for up and down directions to support high-speed cascading in the CoolRunner-II architecture.

The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored; the data outputs (Q) go to logic level zero, terminal count outputs TCU and TCD go to zero and one, respectively, and clock enable outputs CEOU and CEOD go to Low and High, respectively, on the Low-to-High and High-to-Low clock (C) transition. The data on the D inputs loads into the counter on the Low-to-High and High-to-Low clock (C) transition when the load enable input (L) is High, independent of the CE inputs.

All Q outputs increment when CEU is High, provided R and L are Low during the Low-to-High and High-to-Low clock transition. All Q outputs decrement when CED is High, provided R and L are Low. The counter ignores clock transitions when CEU and CED are Low. Both CEU and CED should not be High during the same clock transition; the CEOU and CEOD outputs might not function properly for cascading when CEU and CED are both High.

For counting up, the CEOU output is High when all Q outputs and CEU are High. For counting down, the CED output is High when all Q outputs are Low and CED is High. To cascade counters, the CEOU and CED outputs of each counter are, respectively, connected directly to the CEU and CED inputs of the next stage. The C, L, and R inputs are connected in parallel.

In CoolRunner-II, the maximum clocking frequency of these counter components is unaffected by the number of cascaded stages for all counting and loading functions. The TCU terminal count output is High when all Q outputs are High, regardless of CEU. The TCD output is High when all Q outputs are Low, regardless of CED.

When cascading counters, the final terminal count signals can be produced by AND wiring all the TCU outputs (for the up direction) and all the TCD outputs (for the down direction). The TCU, CEOU, and CEOD outputs are produced by optimizable
AND gates within the component. This results in zero propagation from the CEU and CED inputs and from the Q outputs, provided all connections from each such output remain on-chip. Otherwise, a macrocell buffer delay is introduced.

The counter is initialized to zero (TCU Low and TCD High) when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 1 for CBD2X2; z = 3 for CBD4X2; z = 7 for CBD8X2; z = 15 for CBD16X2
TCU = Qz • Q(z-1) • Q(z-2) • ... • Q0
TCD = Qz • Q(z-1) • Q(z-2) • ... • Q0
CEOU = TCU • CEU
CEOD = TCD • CED
CBD4X2 Implementation CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of cbd2x2 is

signal CEU_INT : std_logic;
signal CED_INT : std_logic;

constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) :=
(others => '1');
constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) :=
(others => '0');

begin

process(C)
begin
if (C'event) then
if (R='1') then
Q <= (others => '0');
CEU_INT <= '0';
CED_INT <= '1';
elsif (L = '1') then
Q <= D;
CEU_INT <= CEU;
CED_INT <= CED;
elsif (CEU='1') then
Q <= Q+1;
CEU_INT <= '1';
CED_INT <= '0';
elsif (CED='1') then
Q <= Q-1;
CEU_INT <= '0';
CED_INT <= '1';
end if;
end if;
end process;

process(Q, CEU_INT, CED_INT)
begin
if ((Q = TERMINAL_COUNT_UP) and (CEU_INT = '1')) then
TCU_INT <= '1';
TCD_INT <= '0';
elsif ((Q = TERMINAL_COUNT_DOWN) and (CED_INT = '1')) then
TCU <= '0';
TCD <= '1';
else
TCU <= '0';
TCD <= '0';
end if;
end process;

CEOU <= TCU and CEU;
CEOD <= TCD and CED;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
if (R)
begin
  Q <= 0;
  CEU_INT <= 1'b0;
  CED_INT <= 1'b1;
end
else if (L)
begin
  Q <= D;
  CEU_INT <= CEU;
  CED_INT <= CED;
end
else if (CEU)
begin
  Q <= Q + 1;
  CEU_INT <= 1;
  CED_INT <= 0;
end
else if (CED)
begin
  Q <= Q - 1;
  CEU_INT <= 1'b0;
  CED_INT <= 1'b1;
end
end
always @ (Q or CEU_INT or CED_INT)
begin
if (Q == TERMINAL_COUNT_UP && CEU_INT)
begin
  TCU_INT <= 1;
  TCD_INT <= 0;
end
else if ((Q == TERMINAL_COUNT_DOWN) && CED_INT)
begin
  TCU_INT <= 0;
  TCD_INT <= 1;
end
else
begin
  TCU_INT <= 0;
  TCD_INT <= 0;
end
end
always @ (TCU_INT or CEU or TCD_INT or CED)
begin
  CEOU <= TCU_INT && CEU;
  CEOD <= TCD_INT && CED;
  TCU <= TCU_INT;
  TCD <= TCD_INT;
end
CC8CE, CC16CE

8-, 16-Bit Cascadable Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CC8CE, CC16CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CC8CE and CC16CE are, respectively, 8- and 16-bit (stage), asynchronous clearable, cascadable binary counters. These counters are implemented using carry logic with relative location constraints to ensure efficient placement of logic. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, with Low outputs, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>C</td>
<td>Qz – Q0</td>
</tr>
<tr>
<td>TC</td>
<td>CEO</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>No Chg</td>
</tr>
</tbody>
</table>
Inputs

<table>
<thead>
<tr>
<th>CLR</th>
<th>CE</th>
<th>C</th>
<th>Qz – Q0</th>
<th>TC</th>
<th>CEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>Inc</td>
<td>TC</td>
<td>CEO</td>
</tr>
</tbody>
</table>

$z = 7$ for CC8CE; $z = 15$ for CC16CE

TC = $Qz \cdot Q(z-1) \cdot Q(z-2) \cdot \ldots \cdot Q0$

CEO = TC $\cdot$ CE

CC8CE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
CC8CE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of cc8ce is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');

begin

  process(C, CLR)
  begin
    if (CLR='1') then
      Q <= (others => '0');
    elsif (C'event and C='1') then
      if (CE='1') then
        Q <= Q+1;
      end if;
    end if;
  end process;

  process (Q)
  begin
    if (Q = TERMINAL_COUNT) then
      TC <= '1';
    else
      TC <= '0';
    end if;
  end process;

  CEO <= TC and CE;

end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    Q <= Q + 1;
end

always @ (Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end
```
always @ (TC or CE)
begin
  CEO <= TC & CE;
end
CC8CLE, CC16CLE

8-, 16-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CC8CLE, CC16CLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CC8CLE and CC16CLE are, respectively, 8- and 16-bit (stage), synchronously loadable, asynchronously clearable, cascadable binary counters. These counters are implemented using carry logic with relative location constraints to ensure efficient placement of logic.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The Q outputs increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the C, L, and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, with Low output, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.
GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 7 for CC8CLE; z = 15 for CC16CLE

TC = Qz・Q(z-1)・Q(z-2)・…・Q0

CEO = TC・CE
CC8CLE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
CC8CLE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of cc8cle is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
  (others => '1');

begin

process(C, CLR)
begin
  if (CLR='1') then
    Q <= (others => '0');
  elsif (C'event and C='1') then
    if (L = '1') then
      Q <= D;
    elsif (CE='1') then
      Q <= Q+1;
    end if;
  end if;
end process;

process(Q)
begin
  if (Q = TERMINAL_COUNT) then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO <= TC and CE;

end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    Q <= Q + 1;
end

always @ (Q)
begin
  if (Q == TERMINAL_COUNT)
    TC <= 1;
  else
    TC <= 0;
end
```
always @ (TC or CE)
beg
 CEO <= TC & CE;
end
CC8CLED, CC16CLED

8-, 16-Bit Loadable Cascadable Bidirectional Binary Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CC8CLED, CC16CLED</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

CC8CLED and CC16CLED are, respectively, 8- and 16-bit (stage), synchronously loadable, asynchronously clearable, cascadable, bidirectional binary counters. These counters are implemented using carry logic with relative location constraints, which assures most efficient logic placement.

The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition, independent of the state of clock enable (CE). The Q outputs decrement when CE is High and UP is Low during the Low-to-High clock transition. The Q outputs increment when CE and UP are High. The counter ignores clock transitions when CE is Low.

For counting up, the TC output is High when all Q outputs and UP are High. For counting down, the TC output is High when all Q outputs and UP are Low. To cascade counters, the count enable out (CEO) output of each counter is connected to the CE pin of the next stage. The clock, UP, L, and CLR inputs are connected in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, outputs Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.
### Inputs

<table>
<thead>
<tr>
<th>CLR</th>
<th>L</th>
<th>CE</th>
<th>C</th>
<th>UP</th>
<th>Dz – D0</th>
<th>Qz – Q0</th>
<th>TC</th>
<th>CEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>↑</td>
<td>X</td>
<td>Dn</td>
<td>Dn</td>
<td>TC</td>
<td>CEO</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No Chg</td>
<td>No Chg</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>X</td>
<td>Inc</td>
<td>TC</td>
<td>CEO</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>0</td>
<td>X</td>
<td>Dec</td>
<td>TC</td>
<td>CEO</td>
</tr>
</tbody>
</table>

*z = 7 for CC8CLED; z = 15 for CC16CLED*

\[
TC = (Qz\cdot Q(z-1)\cdot Q(z-2)\cdot ...\cdot Q0\cdot UP) + (Qz\cdot Q(z-1)\cdot Q(z-2)\cdot ...\cdot Q0\cdot UP) + (Qz\cdot Q(z-1)\cdot Q(z-2)\cdot ...\cdot Q0\cdot UP) + (Qz\cdot Q(z-1)\cdot Q(z-2)\cdot ...\cdot Q0\cdot UP)
\]

CEO = TC • CE
CC8CLED Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
CC8CLED Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage
For HDL, these design elements are inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of cc8cled is

constant TERMINAL_COUNT_UP : std_logic_vector(WIDTH-1 downto 0) :=
(others => '1');
constant TERMINAL_COUNT_DOWN : std_logic_vector(WIDTH-1 downto 0) :=
(others => '0');

begin

process(C, CLR)
begin
if (CLR='1') then
 Q <= (others => '0');
elsif (C'event and C='1') then
 if (L = '1') then
  Q <= D;
 elsif (CE='1') then
  if (UP='1') then
   Q <= Q+1;
   elsif (UP='0') then
    Q <= Q-1;
   end if;
   end if;
   end if;
end if;
end process;

process(Q, UP)
begin
if (((Q = TERMINAL_COUNT_UP) and (UP = '1')) or
 (Q = TERMINAL_COUNT_DOWN) and (UP = '0'))) then
 TC<='1';
 else
 TC<='0';
 end if;
end process;

CEO<='TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
if (CLR)
 Q <= 0;
else if (L)
 Q <= D;
else if (CE)
begin
 if (UP)
  Q <= Q + 1;
 else if (!UP)
  Q <= Q - 1;
always @ (Q or UP)
begin
  if ((Q == TERMINAL_COUNT_UP && UP) || (Q == TERMINAL_COUNT_DOWN && !UP))
    TC <= 1;
  else
    TC <= 0;
end

always @ (TC or CE)
begin
  CEO <= TC & CE;
end
CC8RE, CC16RE

8-, 16-Bit Cascadable Binary Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>CC8RE, CC16RE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-II, Macro</td>
<td></td>
</tr>
<tr>
<td>Spartan-3, Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E, Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X, Macro</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL, No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3, No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II, No</td>
<td></td>
</tr>
</tbody>
</table>

CC8RE and CC16RE are, respectively, 8- and 16-bit (stage), synchronous resettable, cascadable binary counters. These counters are implemented using carry logic with relative location constraints to ensure efficient placement of logic. The synchronous reset (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero on the Low-to-High clock (C) transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs and CE are High.

Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C and R inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, with Low outputs, when power is applied. Spartan-II, Spartan-II, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 7 for CC8RE; z = 15 for CC16RE

TC = Qz * Q(z-1) * Q(z-2) * ... * Q0 * CE

CEO = TC * CE
CC8RE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
CC8RE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of cc8re is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
    (others => '1');

begin

process(C, R)
begin
    if (C'event and C='1') then
        if (R='1') then
            Q <= (others => '0');
        elsif (CE='1') then
            Q <= Q+1;
        end if;
    end if;
end process;

process(Q)
begin
    if (Q = TERMINAL_COUNT) then
        TC<='1';
    else
        TC<='0';
    end if;
end process;

CEO<=TC and CE;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
    if (R)
        Q <= 0;
    else if (CE)
        Q <= Q + 1;
end

always @(Q)
begin
    if (Q == TERMINAL_COUNT)
        TC <= 1;
    else
        TC <= 0;
end
always @(TC or CE)
begin
    CEO <= TC & CE;
end
CD4CE

4-Bit Cascadable BCD Counter with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CD4CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CD4CE is a 4-bit (stage), asynchronous clearable, cascadable binary-coded-decimal (BCD) counter. The asynchronous clear input (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The Q outputs increment when clock enable (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X as shown in the following state diagram. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the counter resets to zero or recovers within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the CLR and clock inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage.
When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse to the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

TC = Q3•Q2•Q1•Q0
CEO = TC•CE

CD4CE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element can be inferred.

VHDL Inference Code

```vhdl
architecture Behavioral of cd4ce is
begin

process (C, CLR)
begin
  if (CLR = '1') then
    Q <= "0000";
  elsif (C'event and C = '1') then
    if (CE = '1') then
      if (Q = "1001") then
        Q <= "0000";
      elsif (Q = "1011") then
        Q <= "0110";
      elsif (Q = "1101") then
        Q <= "0100";
      elsif (Q = "1111") then
        Q <= "0010";
      else
        Q <= Q + 1;
    end if;
  end if;
end process;
```

Libraries Guide
ISE 6.li
www.xilinx.com
1-800-255-7778
297
end if;
end if;
end if;
end process;

process (Q)
begin
if (Q = "1001") then
  TC_INT <= '1';
else
  TC_INT <= '0';
end if;
end process;

TC <= TC;
end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
      else if (Q == 4'b1011)
        Q <= 4'b0110;
      else if (Q == 4'b1101)
        Q <= 4'b0100;
      else if (Q == 4'b1111)
        Q <= 4'b0010;
      else
        Q <= Q + 1;
    end
end

always @ (Q)
begin
  if (Q == 4'b1001)
    TC <= 1;
  else
    TC <= 0;
end

always @ (TC or CE)
begin
  CEO <= TC && CE;
end
CD4CLE is a 4-bit (stage), synchronously loadable, asynchronously clearable, binary-coded-decimal (BCD) counter. The asynchronous clear input (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition. The Q outputs increment when clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X as shown in the following state diagram.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the counter resets to zero or recovers within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the CLR, L, and C inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the
clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

\[ TC = Q3' \cdot Q2' \cdot Q1 \cdot Q0 \]

\[ CEO = TC \cdot CE \]

CD4CLE Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
CD4CLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element can be inferred.

VHDL Inference Code

architecture Behavioral of cd4cle is

begin

process (C, CLR)
begin
if (CLR = '1') then
  Q <= "0000";
elsif (C'event and C = '1') then
  if (L = '1') then
    Q <= D;
  elsif (CE = '1') then
    if (Q = "1001") then
      Q <= "0000";
    elsif (Q = "1011") then
      Q <= "0110";
    elsif (Q = "1101") then
      Q <= "0100";
    elsif (Q = "1111") then
      Q <= "0101";
      Q <= "0100";
    elsif (Q = "1111") then
      Q <= "0100";
    else
      Q <= "0000";
    end if;
  else
    Q <= "0000";
  end if;
elsif (CE = '1') then
  if (Q = "1001") then
    Q <= "0000";
  elsif (Q = "1011") then
    Q <= "0110";
  elsif (Q = "1101") then
    Q <= "0100";
  elsif (Q = "1111") then
    Q <= "0101";
  else
    Q <= "0000";
  end if;
else
  Q <= "0000";
end if;
end process;
end Behavioral;
always @ (posedge C or posedge CLR) 
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
      else if (Q == 4'b1011)
        Q <= 4'b0110;
      else if (Q == 4'b1101)
        Q <= 4'b0100;
      else if (Q == 4'b1111)
        Q <= 4'b0010;
      else
        Q <= Q + 1;
    end
end

always @ (Q)
begin
  if (Q == 4'b1001)
    TC <= 1;
  else
    TC <= 0;
end

always @(TC or CE)
begin
  CEO <= TC && CE;
end
CD4RE

4-Bit Cascadable BCD Counter with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>CD4RE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CD4RE is a 4-bit (stage), synchronous resettable, cascadable binary-coded-decimal (BCD) counter. The synchronous reset input (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero on the Low-to-High clock (C) transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X as shown in the following state diagram. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the counter resets to zero or recovers within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the R and clock inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When
cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

TC = Q3•Q2•Q1•Q0
CEO = TC•CE

**CD4RE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II**

**Usage**

For HDL, this design element can be inferred.

**VHDL Inference Code**

```vhdl
architecture Behavioral of cd4re is
begin
    process (C)
    begin
        if (C'event and C = '1') then
            if (R = '1') then
                Q <= "0000";
            elsif (CE = '1') then
                if (Q = "1001") then
                    Q <= "0000";
                elsif (Q = "1011") then
                    Q <= "0110";
                elsif (Q = "1101") then
                    Q <= "0100";
                elsif (Q = "1111") then
                    Q <= "0010";
                else
                    Q <= Q + 1;
                end if;
            end if;
        end if;
    end process C;
end architecture;
```
end if;
end process;

process (Q)
begin
if (Q = "1001") then
  TC <= '1';
else
  TC <= '0';
end if;
end process;

CEO <= TC and CE;
end Behavioral;

Verilog Inference Code
always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
      else if (Q == 4'b1011)
        Q <= 4'b0110;
      else if (Q == 4'b1101)
        Q <= 4'b0100;
      else if (Q == 4'b1111)
        Q <= 4'b0010;
      else
        Q <= Q + 1;
    end
end

always @ (Q)
begin
  if (Q == 4'b1001)
    TC <= 1;
  else
    TC <= 0;
end

always @(TC or CE)
begin
  CEO <= TC && CE;
end
CD4RLE

4-Bit Loadable Cascadable BCD Counter with Clock Enable and Synchronous Reset

**Architectures Supported**

<table>
<thead>
<tr>
<th>CD4RLE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CD4RLE is a 4-bit (stage), synchronous loadable, resettable, binary-coded-decimal (BCD) counter. The synchronous reset input (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero on the Low-to-High clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock (C) transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low.

The counter recovers from any of six possible illegal states and returns to a normal count sequence within two clock cycles for Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X as shown in the following state diagram. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the counter resets to zero or recovers within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the R, L, and C inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When
cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/ XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TC = Q3•Q2•Q1•Q0
CEO = TC•CE
CD4RLE Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
CD4RLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

architecture Behavioral of cd4rle is

begin

process (C)
begin
  if (C'event and C = '1') then
    if (R = '1') then
      Q <= "0000";
    elsif (L = '1') then
      Q <= D;
    elsif (CE = '1') then
      if (Q = "1001") then
        Q <= "0000";
      elsif (Q = "1011") then
        Q <= "0110";
      elsif (Q = "1101") then
        Q <= "0100";
      elsif (Q = "1111") then
        Q <= "0010";
      else
        Q <= Q + 1;
      end if;
    end if;
  end if;
end process;

process (Q)
begin
  if (Q = "1001") then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO <= TC and CE;

end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
    end if;
  end if;
end
else if (Q == 4'b1011)
    Q <= 4'b0110;
else if (Q == 4'b1101)
    Q <= 4'b0100;
else if (Q == 4'b1111)
    Q <= 4'b0010;
else
    Q <= Q + 1;
end
end

always @(Q)
begin
    if (Q == 4'b1001)
        TC <= 1;
    else
        TC <= 0;
end

always @(TC or CE)
begin
    CEO <= TC & CE;
end
CDD4CE

4-Bit Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CDD4CE</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CDD4CE is a 4-bit (stage), asynchronous clearable, cascadable dual edge triggered Binary-coded-decimal (BCD) counter. The asynchronous clear input (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The Q outputs increment when clock enable (CE) is High during the Low-to-High and High-to-Low clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low. The counter recovers to zero from any illegal state within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the CLR and clock inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse to the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
CDD4CE Implementation CoolRunner-II

Usage

For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

architecture Behavioral of cdd4ce is

begin

    process (C, CLR)
    begin
        if (CLR = '1') then
            Q <= "0000";
        elsif (C'event) then
            if (CE = '1') then
                if (Q = "1001") then
                    Q <= "0000";
                elsif (Q = "1011") then
                    Q <= "0110";
                end if;
            end if;
        end if;
    end process;
end;
elsif (Q = "1101") then
  Q <= "0100";
elsif (Q = "1111") then
  Q <= "0010";
else
  Q <= Q + 1;
end if;
end if;
end if;
end process;

process (Q)
begin
  if (Q = "1001") then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO <= TC and CE;
end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
      else if (Q == 4'b1011)
        Q <= 4'b0110;
      else if (Q == 4'b1101)
        Q <= 4'b0100;
      else if (Q == 4'b1111)
        Q <= 4'b0010;
      else
        Q <= Q + 1;
    end
  end
always @(Q)
begin
  if (Q == 4'b1001)
    TC <= 1;
  else
    TC <= 0;
end
always @(TC or CE)
begin
  CEO <= TC && CE;
end
CDD4CLE

4-Bit Loadable Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CDD4CLE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CDD4CLE is a 4-bit (stage), synchronously loadable, asynchronously clearable, dual edge triggered Binary-coded-decimal (BCD) counter. The asynchronous clear input (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High and High-to-Low clock (C) transitions. The Q outputs increment when clock enable input (CE) is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low. The counter recovers to zero from any illegal state within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the CLR, L, and C inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR L CE D3 – D0 C</td>
<td>Q3 Q2 Q1 Q0 TC CEO</td>
</tr>
<tr>
<td>1 X X X X</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 1 X D3 – D0 ↑</td>
<td>D3 D2 D1 D0 TC CEO</td>
</tr>
<tr>
<td>0 1 X D3 – D0 ↓</td>
<td>D3 D2 D1 D0 TC CEO</td>
</tr>
<tr>
<td>0 0 1 X ↑</td>
<td>Inc Inc Inc Inc TC CEO</td>
</tr>
</tbody>
</table>
CDD4CLE Implementation CoolRunner-II

Usage

For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of cdd4cle is
begin
  process (C, CLR)
  begin
    if (CLR = '1') then
```

Q <= "0000";
elif (C'event) then
  if (L = '1') then
    Q <= D;
elif (CE = '1') then
  if (Q = "1001") then
    Q <= "0000";
elif (Q = "1011") then
    Q <= "0110";
elif (Q = "1101") then
    Q <= "0100";
elif (Q = "1111") then
    Q <= "0010";
else
    Q <= Q + 1;
    end if;
  end if;
  end if;
end if;
end process;

process (Q)
begin
  if (Q = "1001") then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;
CEO <= TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
      else if (Q == 4'b1011)
        Q <= 4'b0110;
      else if (Q == 4'b1101)
        Q <= 4'b0100;
      else if (Q == 4'b1111)
        Q <= 4'b0010;
      else
        Q <= Q + 1;
    end
end
always @ (Q)
begin
if (Q == 4'b1001)
    TC <= 1;
else
    TC <= 0;
end

always @(TC or CE)
begin
    CEO <= TC && CE;
end
CDD4RE

4-Bit Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>Architectures Supported</th>
<th>CDD4RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CDD4RE is a 4-bit (stage), synchronous resettable, cascadable dual edge triggered binary-coded-decimal (BCD) counter. The synchronous reset input (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero on the Low-to-High or High-to-Low clock (C) transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High and High-to-Low clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low. The counter recovers to zero from any illegal state within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the R and clock inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
CDD4RE Implementation CoolRunner-II

Usage
For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

architecture Behavioral of cdd4re is

begin

  process (C)
  begin
    if (C'event) then
      if (R = '1') then
        Q <= "0000";
      elsif (CE = '1') then
        if (Q = "1001") then
          Q <= "0000";
        end if;
      end if;
    end if;
  end process;

  if (Q(3)) then
    Q <= "1001";
  end if;

end Behavioral;
elsif (Q = "1011") then
  Q <= "0110";
elsif (Q = "1101") then
  Q <= "0100";
elsif (Q = "1111") then
  Q <= "0010";
else
  Q <= Q + 1;
  end if;
end if;
end if;
end process;

process (Q)
begin
  if (Q = "1001") then
    TC <= '1';
  else
    TC <= '0';
  end if;
end process;

CEO <= TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
  if (R)
    Q <= 0;
  else if (CE)
    begin
      if (Q == 4'b1001)
        Q <= 4'b0000;
      else if (Q == 4'b1011)
        Q <= 4'b0110;
      else if (Q == 4'b1101)
        Q <= 4'b0100;
      else if (Q == 4'b1111)
        Q <= 4'b0010;
      else
        Q <= Q + 1;
    end
  end

always @ (Q)
begin
  if (Q == 4'b1001)
    TC <= 1;
  else
    TC <= 0;
end
always @ (TC or CE)
begin
    CEO <= TC && CE;
end
CDD4RLE

4-Bit Loadable Cascadable Dual Edge Triggered BCD Counter with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>CDD4RLE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CDD4RLE is a 4-bit (stage), synchronous loadable, resettable, dual edge triggered binary-coded-decimal (BCD) counter. The synchronous reset input (R) is the highest priority input. When R is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero on the Low-to-High or High-to-Low clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High and High-to-Low clock (C) transition. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High and High-to-Low clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when Q3 and Q0 are High and Q2 and Q1 are Low. The counter recovers to zero from any illegal state within the first clock cycle.

Larger counters are created by connecting the count enable out (CEO) output of the first stage to the CE input of the next stage and connecting the R, L, and C inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than \( n(t_{CE-TC}) \), where \( n \) is the number of stages and the time \( t_{CE-TC} \) is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R L CE D3 – D0 C</td>
<td>Q3 Q2 Q1 Q0 TC CEO</td>
</tr>
<tr>
<td>1 X X X ↑</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 X X X ↓</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 1 X D3 – D0 ↑</td>
<td>D3 D2 D1 D0 TC CEO</td>
</tr>
<tr>
<td>0 1 X D3 – D0 ↓</td>
<td>D3 D2 D1 D0 TC CEO</td>
</tr>
<tr>
<td>0 0 1 X ↑</td>
<td>Inc Inc Inc Inc TC CEO</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TC = Q3•!Q2•!Q1•Q0
CEO = TC•CE
CDD4RLE Implementation CoolRunner-II

Usage

For HDL, this design element can be inferred but not instantiated.
VHDL Inference Code

architecture Behavioral of cdd4rle is

begin

process (C)
begin
if (C'event) then
if (R = '1') then
Q <= "0000";
elssif (L = '1') then
Q <= D;
elssif (CE = '1') then
if (Q = "1001") then
Q <= "0000";
elssif (Q = "1011") then
Q <= "0110";
elssif (Q = "1101") then
Q <= "0100";
elssif (Q = "1111") then
Q <= "0010";
else
Q <= Q + 1;
end if;
end if;
end if;
en process;

process (Q)
begin
if (Q = "1001") then
TC <= '1';
elssif (Q = "1011") then
TC <= '0';
else
TC <= '0';
end if;
en process;

CEO <= TC and CE;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
if (R)
Q <= 0;
elssif (L)
Q <= D;
elssif (CE)
begin
if (Q == 4'b1001)
Q <= 4'b0000;
elssif (Q == 4'b1011)
Q <= 4'b0110;
elssif (Q == 4'b1111)
Q <= 4'b0010;
else if (Q == 4'b1111)
  Q <= 4'b0010;
else
  Q <= Q + 1;
end
end

always @(Q)
begin
  if (Q == 4'b1001)
    TC <= 1;
  else
    TC <= 0;
end

always @(TC or CE)
begin
  CEO <= TC && CE;
end
CJ4CE, CJ5CE, CJ8CE

4-, 5-, 8-Bit Johnson Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CJ4CE, CJ5CE, CJ8CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CJ4CE, CJ5CE, and CJ8CE are clearable Johnson/shift counters. The asynchronous clear (CLR) input, when High, overrides all other inputs and causes the data (Q) outputs to go to logic level zero, independent of clock (C) transitions. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low.

For CJ4CE, the Q3 output is inverted and fed back to input Q0 to provide continuous counting operation. For CJ5CE, the Q4 output is inverted and fed back to input Q0. For CJ8CE, the Q7 output is inverted and fed back to input Q0.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

CJ4CE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition
**CJ5CE Truth Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

**CJ8CE Truth Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

**CJ8CE Implementation**


**Usage**

For HDL, this design element can be inferred but not instantiated.
VHDL Inference Code

architecture Behavioral of cj4ce is

begin

process (C, CLR)
begin
if (CLR = '1') then
Q <= (others => '0');
elself (C'event and C = '1') then
if (CE = '1') then
Q(0) <= not Q(WIDTH-1);
Q(WIDTH-1 downto 1) <= Q(WIDTH-2 downto 0);
end if;
elself end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
if (CLR)
Q <= 0;
elself
begin
if (CE)
begin
Q[0] <= !Q[WIDTH-1];
Q[WIDTH-1:1] <= Q[WIDTH-2:0];
end
end
end
CJ4RE, CJ5RE, CJ8RE

4-, 5-, 8-Bit Johnson Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>CJ4RE, CJ5RE, CJ8RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CJ4RE, CJ5RE, and CJ8RE are resettable Johnson/shift counters. The synchronous reset (R) input, when High, overrides all other inputs and causes the data (Q) outputs to go to logic level zero during the Low-to-High clock (C) transition. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High clock transition. Clock transitions are ignored when CE is Low.

For CJ4RE, the Q3 output is inverted and fed back to input Q0 to provide continuous counting operations. For CJ5RE, the Q4 output is inverted and fed back to input Q0. For CJ8RE, the Q7 output is inverted and fed back to input Q0.

The counter is asynchronously cleared, output Low, when power is applied. For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

CJ4RE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition
CJ5RE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

CJ8RE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

Usage

For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

architecture Behavioral of cj4re is

begin

process (C)
begin
if (C'event and C = '1') then
 if (R = '1') then
  Q <= (others => '0');
elsif (CE = '1') then
  Q(0) <= not Q(WIDTH-1);
  Q(WIDTH-1 downto 1) <= Q(WIDTH-2 downto 0);
end if;
end if;
end process;

end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
 if (R)
  Q <= 0;
 else
  begin
   if (CE)
   begin
    Q[0] <= !Q[WIDTH-1];
    Q[WIDTH-1:1] <= Q[WIDTH-2:0];
   end
  end
end
**CJD4CE, CJD5CE, CJD8CE**

4-, 5-, 8-Bit Dual Edge Triggered Johnson Counters with Clock Enable and Asynchronous Clear

**Architectures Supported**

<table>
<thead>
<tr>
<th>CJD4CE, CJD5CE, CJD8CE</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

CJD4CE, CJD5CE, and CJD8CE are dual edge triggered clearable Johnson/shift counters. The asynchronous clear (CLR) input, when High, overrides all other inputs and causes the data (Q) outputs to go to logic level zero, independent of clock (C) transitions. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High and High-to-Low clock transition. Clock transitions are ignored when CE is Low.

For CJD4CE, the Q3 output is inverted and fed back to input Q0 to provide continuous counting operations. For CJD5CE, the Q4 output is inverted and fed back to input Q0. For CJD8CE, the Q7 output is inverted and fed back to input Q0.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

**CJD4CE Truth Table**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition
CJD5CE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

CJD8CE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

Usage

For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

architecture Behavioral of cjd4ce is

begin

process (C, CLR)
begin
if (CLR = '1') then
Q <= (others => '0');
elsif (C'event) then
if (CE = '1') then
Q(0) <= not Q(WIDTH-1);
Q(WIDTH-1 downto 1) <= Q(WIDTH-2 downto 0);
end if;
end if;
end process;
end Behavioral;
Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
  begin
    if (CLR)
      Q_INT <= 0;
    else
      begin
        if (CE)
          begin
            Q_INT[0] <= !Q_INT[WIDTH-1];
            Q_INT[WIDTH-1:1] <= Q_INT[WIDTH-2:0];
          end
        end
      end
  end

always @ (Q_INT)
  begin
    Q <= Q_INT;
  end
CJD4RE, CJD5RE, CJD8RE

4-, 5-, 8-Bit Dual Edge Triggered Johnson Counters with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>CJD4RE, CJD5RE, CJD8RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CJD4RE, CJD5RE, and CJD8RE are resettable dual edge triggered Johnson/shift counters. The synchronous reset (R) input, when High, overrides all other inputs and causes the data (Q) outputs to go to logic level zero during the Low-to-High and High-to-Low clock (C) transition. The counter increments (shifts Q0 to Q1, Q1 to Q2, and so forth) when the clock enable input (CE) is High during the Low-to-High and High-to-Low clock transition. Clock transitions are ignored when CE is Low.

For CJD4RE, the Q3 output is inverted and fed back to input Q0 to provide continuous counting operations. For CJD5RE, the Q4 output is inverted and fed back to input Q0. For CJD8RE, the Q7 output is inverted and fed back to input Q0.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

CJD4RE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition
CJD5RE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

CJD8RE Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

q = state of referenced output one setup time prior to active clock transition

Usage

For HDL, this design element can be inferred but not instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of cjd4re is
begin
    process (C)
    begin
        if (C'event) then
            if (R = '1') then
                Q <= (others => '0');
            elsif (CE = '1') then
                Q(0) <= not Q(WIDTH-1);
                Q(WIDTH-1 downto 1) <= Q(WIDTH-2 downto 0);
            end if;
        end if;
    end process;
end Behavioral;
```
Verilog Inference Code

```verilog
always @ (posedge C or negedge C)
    begin
        if (R)
            Q <= 0;
        else
            begin
                if (CE)
                    begin
                        Q[0] <= !Q[WIDTH-1];
                        Q[WIDTH-1:1] <= Q[WIDTH-2:0];
                    end
            end
    end
```

```verilog```
Global Clock Divider

Architectures Supported

<table>
<thead>
<tr>
<th>CLK_DIV2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK_DIV4, CLK_DIV6, CLK_DIV8, CLK_DIV10, CLK_DIV12, CLK_DIV14, CLK_DIV16</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

The CLK_DIV2,4,6,8,10,12,14,16 Global Clock Dividers divide a user-provided external clock signal gclk<2> by 2, 4, 6, 8, 10, 12, 14, and 16, respectively. Only one clock divider may be used per design. The global clock divider is available on the XC2C128, XC2C256, XC2C384, and XC2C512 CoolRunner-II devices, but not the XC2C32 or XC2C64. The CLKIN input can only be connected to the device gclk<2> pin. The duty cycle of the CLKDV output is 50-50.

The CLKDV output is reset low by power-on reset circuitry.

Usage

This design element is supported for schematics and instantiation but not for inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CLK_DIVn should be placed
-- after architecture statement but before begin keyword

component CLK_DIVn
  port (CLKDV : out STD_ULOGIC;
        CLKin : in STD_ULOGIC);
end component;

-- Component Instantiation for CLK_DIVn should be placed
-- in architecture after the begin keyword
```
CLK_DIVn_INSTANCE_NAME : CLK_DIVn
  port map (CLKDV => user_CLKDV,
            CLKN => user_CLKN);

Verilog Instantiation Template

CLK_DIVn CLK_DIVn_instance_name (.CLKDV (user_CLKDV),
                                .CLIN (user_CLIN));
Global Clock Divider with Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

The CLK_DIV2,4,6,8,10,12,14,16R Global Clock Dividers with Synchronous Reset divide a user-provided external clock signal gclk<2> by 2, 4, 6, 8, 10, 12, 14, and 16, respectively. Only one clock divider may be used per design. The global clock divider is available on the XC2C128, XC2C256, XC2C384, and XC2C512 CoolRunner-II devices, but not the XC2C32 or XC2C64. The CLKIN and CDRST inputs can only be connected to the device gclk<2> and CDRST pins. The duty cycle of the CLKDV output is 50-50.

The CDRST input is an active High synchronous reset. If CDRST is input High when the CLKDV output is High, the CLKDV output remains High to complete the last clock pulse, and then goes Low.

The CLKDV output is reset low by power-on reset circuitry.

Usage

For HDL, these design elements are supported for instantiation but not inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CLK_DIVnR should be placed
-- after architecture statement but before begin keyword

component CLK_DIVnR
    port (CLDV : out STD_ULOGIC;
         CDRST : in STD_ULOGIC;
```
CLIN : in STD_ULOGIC);
end component;

-- Component Instantiation for CLK_DIVnR should be placed
-- in architecture after the begin keyword

CLK_DIVnR_INSTANCE_NAME : CLK_DIVnR
    port map (CLKDV => user_CLKDV,
              CDRST => user_CDRST,
              CLKIN => user_CLKIN);

Verilog Instantiation Template

CLK_DIVnR CLK_DIVnR_instance_name (.CLKDV (user_CLKDV),
                                 .CDRST (user_CDRST),
                                 .CLKIN (user_CLKIN));
Global Clock Divider with Synchronous Reset and Start Delay

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

The CLK_DIV2,4,6,8,10,12,14,16 Global Clock Dividers with Synchronous Reset and Start Delay divide a user-provided external clock signal gclk<2> by 2, 4, 6, 8, 10, 12, 14, and 16, respectively. Only one clock divider may be used per design. The global clock divider is available on the XC2C128, XC2C256, XC2C384, and XC2C512 CoolRunner-II devices, but not the XC2C32 or XC2C64. The CLKIN and CDRST inputs can only be connected to the device gclk<2> and CDRST pins. The duty cycle of the CLKDV output is 50-50.

The CDRST input is an active High synchronous reset. If CDRST is input High when the CLKDV output is High, the CLKDV output remains High to complete the last clock pulse, and then goes Low.

The start delay function delays the start of the CLKDV output by (n + 1) clocks, where n is the divisor for the clock divider.

The CLKDV output is reset low by power-on reset circuitry.

Usage

For HDL, these design elements are supported for instantiation but not inference.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CLK_DIVnRSD should be placed
-- after architecture statement but before begin keyword
```
component CLK_DIVnRSD
  port (CLDV : out STD_ULOGIC;
       CDRST : in STD_ULOGIC;
       CLKin : in STD_ULOGIC);
end component;

-- Component Instantiation for CLK_DIVnRSD should be placed
-- in architecture after the begin keyword

CLK_DIVnRSD_INSTANCE_NAME : CLK_DIVnRSD
  port map (CLKDV => user_CLKDV,
            CDRST => user_CDRST,
            CLKin => user_CLKin);

Verilog Instantiation Template

CLK_DIVnRSD CLK_DIVnRSD_instance_name (.CLKDV (user_CLKDV),
                                        .CDRST (user_CDRST),
                                        .CLKin (user_CLKin));
The CLK_DIV2,4,6,8,10,12,14,16SD Global Clock Dividers with Start Delay divide a user-provided external clock signal gclk<2> by 2, 4, 6, 8, 10, 12, 14, and 16, respectively. Only one clock divider may be used per design. The global clock divider is available on the XC2C128, XC2C256, XC2C384, and XC2C512 CoolRunner-II devices, but not the XC2C32 or XC2C64. The CLkin input can only be connected to the device gclk<2> pin. The duty cycle of the CLKDV output is 50-50.

The start delay function delays the CLKDV output (n + 1) clocks, where n is the divisor for the clock divider.

The CLKDV output is reset low by power-on reset circuitry.

Usage
This design element is supported for schematics and instantiation but not for inference.

VHDL Instantiation Template

```
-- Component Declaration for CLK_DIVnSD should be placed
-- after architecture statement but before begin keyword

component CLK_DIVnSD
    port (CLKDV : out STD_ULOGIC;
          CLkin : in STD_ULOGIC);
end component;
```
-- Component Instantiation for CLK_DIVnSD should be placed
-- in architecture after the begin keyword

CLK_DIVnSD_INSTANCE_NAME : CLK_DIVnSD
    port map (CLKDV => user_CLKDV,
              CLKIN => user_CL Kin);

Verilog Instantiation Template

CLK_DIVnSD CLK_DIVnSD_instance_name (.CLKDV (user_CLKDV),
                                      .CLKin (user_CL Kin));
CLKDLL

Clock Delay Locked Loop

Architectures Supported

<table>
<thead>
<tr>
<th>CLKDLL</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

* Use CLKDLLLE for Spartan-IIE and Virtex-E.

CLKDLL is a clock delay locked loop used to minimize clock skew. CLKDLL synchronizes the clock signal at the feedback clock input (CLKFB) to the clock signal at the input clock (CLKIN). The locked output (LOCKED) is high when the two signals are in phase. The signals are considered to be in phase when their rising edges are within a specific range of each other (see The Programmable Logic Data Book for the most current value).

The frequency of the clock signal at the CLKIN input must be in a specific range depending on speed grade (see The Programmable Logic Data Book for the most current values). The CLKIN pin must be driven by an IBUF or a BUFG.

On-chip synchronization is achieved by connecting the CLKFB input to a point on the global clock network driven by a BUFG, a global clock buffer. The BUFG connected to the CLKFB input of the CLKDLL must be sourced from either the CLK0 or CLK2X outputs of the same CLKDLL. The CLKIN input should be connected to the output of an IBUF, with the IBUF input connected to a pad driven by the system clock.

Off-chip synchronization is achieved by connecting the CLKFB input to the output of an IBUF, with the IBUF input connected to a pad. Either the CLK0 or CLK2X output can be used but not both. The CLK0 or CLK2X must be connected to the input of OBUF, an output buffer.

The duty cycle of the CLK0 output is 50-50 unless the DUTY_CYCLE_CORRECTION attribute is set to FALSE, in which case the duty cycle is the same as that of the CLKIN input. The duty cycle of the phase shifted outputs (CLK90, CLK180, and CLK270) is the same as that of the CLK0 output. The duty cycle of the CLK2X and CLKDV outputs is always 50-50. The frequency of the CLKDV output is determined by the value assigned to the CLKDV_DIVIDE attribute.

The master reset input (RST) resets CLKDLL to its initial (power-on) state. The signal at the RST input is asynchronous and must be held High for just 2ns.
CLKDLL Outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK0</td>
<td>Clock at 1x CLKin frequency</td>
</tr>
<tr>
<td>CLK180</td>
<td>Clock at 1x CLKin frequency, shifted 180° with regards to CLK0</td>
</tr>
<tr>
<td>CLK270</td>
<td>Clock at 1x CLKin frequency, shifted 270° with regards to CLK0</td>
</tr>
<tr>
<td>CLK2X</td>
<td>Clock at 2x CLKin frequency, in phase with CLK0</td>
</tr>
<tr>
<td>CLK90</td>
<td>Clock at 1x CLKin frequency, shifted 90° with regards to CLK0</td>
</tr>
<tr>
<td>CLKDV</td>
<td>Clock at (1/n)x CLKin frequency, n=CLKDV_DIVIDE value. CLKDV is in phase with CLK0.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>CLKDLL locked</td>
</tr>
</tbody>
</table>

**Note:** See the "PERIOD Specifications on CLKDLLs and DCM" section of the "Xilinx Constraints P" chapter in the *Constraints Guide* for additional information on using the TNM, TNM_NET, and PERIOD attributes with CLKDLL components.

**Usage**

This component is generally instantiated in the code as it can not be easily inferred in synthesis tools. Some synthesis tools may allow inference via an attribute. See your synthesis tool’s documentation. Generally, global buffers (IBUF, BUF) are instantiated with the CLKDLL component to construct the proper clocking circuit. See the XAPP 132 application note, "Using the Virtex Delay-Locked Loop" and the *Xilinx Databook* for more information on using the CLKDLL component.

**VHDL Instantiation Template**

-- Component Declaration for CLKDLL should be placed
-- after architecture statement but before begin keyword

component CLKDLL

    -- synthesis translate_off

    generic (CLKDV_DIVIDE : real := 2.0; -- (1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 8.0, 16.0)
              DUTY_CYCLE_CORRECTION : Boolean := TRUE; -- (TRUE, FALSE)
              STARTUP_WAIT : boolean := FALSE); -- (TRUE, FALSE)

    -- synthesis translate_on

    port (CLK0 : out STD_ULOGIC;
          CLK180 : out STD_ULOGIC;
          CLK270 : out STD_ULOGIC;
          CLK2X : out STD_ULOGIC;
          CLK90 : out STD_ULOGIC;
          CLKDV : out STD_ULOGIC;
          LOCKED : out STD_ULOGIC;
CLKFB : in STD_ULOGIC;
CLKin : in STD_ULOGIC;
RST : in STD_ULOGIC);
end component;

-- Component Attribute specification for CLKDLL
-- should be placed after architecture declaration but
-- before the begin keyword

attribute CLKDV_DIVIDE : real;
attribute DUTY_CYCLE_CORRECTION : boolean;
attribute STARTUP_WAIT : boolean;

attribute CLKDV_DIVIDE of CLKDLL_instance_name: label is 2.0;
-- 1.5,2,2.5,3,4, 5, 8, 16 are valid for CLKDV_DIVIDE
attribute DUTY_CYCLE_CORRECTION of CLKDLL_instance_name: label is "TRUE";
-- TRUE, FALSE are valid for DUTY_CYCLE_CORRECTION
attribute STARTUP_WAIT of CLKDLL_instance_name: label is "FALSE";--
(TRUE,FALSE)

-- Component Instantiation for CLKDLL should be placed
-- in architecture after the begin keyword

CLKDLL_INSTANCE_NAME : CLKDLL
-- synthesis translate_off
  generic map (CLKDV_DIVIDE => real_value, -- (1.5,2,2.5,3,4,5,8,16)
    DUTY_CYCLE_CORRECTION => boolean_value, -- (TRUE, FALSE)
    STARTUP_WAIT => boolean_value); -- (TRUE, FALSE)
-- synthesis translate_on
port map (CLK0 => userCLK0,
  CLK180 => userCLK180,
  CLK270 => userCLK270,
  CLK2X => userCLK2X,
  CLK90 => userCLK90,
  CLKDV => userCLKDV,
LOCKED => user_LOCKED,
CLKFB => userCLKFB,
CLKIN0 => userCLKIN,
RST => user_RST);

Verilog Instantiation Template

CLKDLL CLKDLL_instance_name (.CLK0 (user_CLK0),
  .CLK180 (user_CLK180),
  .CLK270 (user_CLK270),
  .CLK2X (user_CLK2X),
  .CLK90 (user_CLK90),
  .CLKDV (user_CLKDV),
  .LOCKED (user_LOCKED),
  .CLKFB (user_CLKFB),
  .CLKIN (user_CLKIN),
  .RST (user_RST));

defparam CLKDLL_instance_name.CLKDV_DIVIDE = integer_value;
  // (1.5,2,2.5,3,4,5,8,16)
defparam CLKDLL_instance_name.DUTY_CYCLE_CORRECTION = boolean_value; //
  (TRUE, FALSE)
defparam CLKDLL_instance_name.STARTUP_WAIT = boolean_value; // (TRUE, FALSE)

Note: Additional syntax may be necessary in order to pass the CLKDLL attributes via the
  synthesis tool. The above defparam statements may need to be isolated from the synthesis tool
  with translate_off/translate_on directives. See your synthesis tool documentation for more
  information on Verilog attribute passing to ensure that you properly pass these attributes to the
  synthesis tool. Otherwise, you may pass these attributes to the UCF file.

Commonly Used Constraints

STARTUP_WAIT, DUTY_CYCLE_CORRECTION, CLKDV_DIVIDE and LOC.
Virtex-E Clock Delay Locked Loop

Architectures Supported

<table>
<thead>
<tr>
<th>CLKDLLE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

* Supported for Spartan-IIE and Virtex-E devices only.

CLKDLLE is a clock delay locked loop used to minimize clock skew for Virtex-E devices. CLKDLLE synchronizes the clock signal at the feedback clock input (CLKFB) to the clock signal at the input clock (CLKIN). The locked output (LOCKED) is high when the two signals are in phase. The signals are considered to be in phase when their rising edges are within a specific range of each other (see The Programmable Logic Data Book for the most current value).

The frequency of the clock signal at the CLkin input must be in a specific range depending on speed grade (see The Programmable Logic Data Book for the most current values). The CLKin pin must be driven by an IBUF or a BUFG.

On-chip synchronization is achieved by connecting the CLKFB input to a point on the global clock network driven by a BUFG, a global clock buffer. The BUFG input can only be connected to the CLK0 or CLK2X output of CLKDLLE. The BUFG connected to the CLKFB input of the CLKDLLE must be sourced from either the CLK0 or CLK2X outputs of the same CLKDLLE. The CLKin input should be connected to the output of an IBUF, with the IBUF input connected to a pad driven by the system clock.

Off-chip synchronization is achieved by connecting the CLKFB input to the output of an IBUF, with the IBUF input connected to a pad. Either the CLK0 or CLK2X output can be used but not both. The CLK0 or CLK2X must be connected to the input of OBUF, an output buffer.

The duty cycle of the CLK0 output is 50-50 unless the DUTY_CYCLE_CORRECTION attribute is set to FALSE, in which case the duty cycle is the same as that of the CLKin input. The duty cycle of the phase shifted outputs (CLK90, CLK180, and CLK270) is the same as that of the CLK0 output. The duty cycle of the CLK2X, CLK2X180, and CLKDV outputs is always 50-50. The frequency of the CLKDV output is determined by the value assigned to the CLKDV_DIVIDE attribute.

The master reset input (RST) resets CLKDLLE to its initial (power-on) state. The signal at the RST input is asynchronous and must be held High for just 2ns.
CLKDLLE Outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK0</td>
<td>Clock at 1x CLkin frequency</td>
</tr>
<tr>
<td>CLK180</td>
<td>Clock at 1x CLK0 frequency, shifted 180° with regards to CLK0</td>
</tr>
<tr>
<td>CLK270</td>
<td>Clock at 1x CLK0 frequency, shifted 270° with regards to CLK0</td>
</tr>
<tr>
<td>CLK2X</td>
<td>Clock at 2x CLK0 frequency, in phase with CLK0</td>
</tr>
<tr>
<td>CLK2X180</td>
<td>Clock at 1x CLK2X frequency shifted 180° with regards to CLK2X</td>
</tr>
<tr>
<td>CLK90</td>
<td>Clock at 1x CLK0 frequency, shifted 90° with regards to CLK0</td>
</tr>
<tr>
<td>CLKDV</td>
<td>Clock at (1/n) x CLK0 frequency, where n=CLKDV_DIVIDE value. CLKDV is in phase with CLK0.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>CLKDLLE locked. CLkin and CLKFB synchronized.</td>
</tr>
</tbody>
</table>

Usage

This component is generally instantiated in the code as it cannot be easily inferred in synthesis tools. Some synthesis tools may allow inference via an attribute. See your synthesis tool documentation. Generally, global buffers (IBUF, BUF) are instantiated with the CLKDLLE component to construct the proper clocking circuit. See the XAPP 132 application note, “Using the Virtex Delay-Locked Loop” and the Xilinx Databook for more information on using the CLKDLLE component.

VHDL Instantiation Template

```vhdl
-- Component Declaration for CLKDLLE should be placed
-- after architecture statement but before begin keyword

component CLKDLLE
  -- synthesis translate_off
  generic (CLKDV_DIVIDE : real := 2.0; -- (1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.5, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0)
  DUTY_CYCLE_CORRECTION : Boolean := TRUE; -- (TRUE, FALSE)
  STARTUP_WAIT :boolean := FALSE); -- (TRUE, FALSE)
  -- synthesis translate_on
  port (CLK0 : out STD_ULOGIC;
      CLK180 : out STD_ULOGIC;
      CLK270 : out STD_ULOGIC;
      CLK2X : out STD_ULOGIC;
      CLK2X180: out STD_ULOGIC;
      CLK90 : out STD_ULOGIC;
      CLKDV : out STD_ULOGIC;
      LOCKED : out STD_ULOGIC;
      CLKFB : in STD_ULOGIC;
      CLkin : in STD_ULOGIC;
      RSt : in STD_ULOGIC);
end component;

-- Component Attribute specification for CLKDLLE
-- should be placed after architecture declaration but
-- before the begin keyword
attribute CLKDV_DIVIDE : real;
```

attribute DUTY_CYCLE_CORRECTION : boolean;
attribute STARTUP_WAIT : boolean;

attribute CLKDV_DIVIDE of CLKDLLE_instance_name: label is 2.0;
-- (1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.5, 8.0, 9.0, 10.0, 11.0,
-- 12.0, 13.0, 14.0, 15.0, 16.0) are valid for CLKDV_DIVIDE
attribute DUTY_CYCLE_CORRECTION of CLKDLLE_instance_name: label is TRUE;
-- (TRUE, FALSE) are valid for DUTY_CYCLE_CORRECTION
attribute STARTUP_WAIT of CLKDLLE_instance_name: label is FALSE; -- (TRUE, FALSE)

-- Component Instantiation for CLKDLLE should be placed
-- in architecture after the begin keyword

CLKDLLE_INSTANCE_NAME : CLKDLLE
-- synthesis translate_off
generic map (CLKDV_DIVIDE => real_value, -- (1.5,2,2.5,3,4,5,8,16)
DUTY_CYCLE_CORRECTION => boolean_value, -- (TRUE, FALSE)
STARTUP_WAIT => boolean_value); -- (TRUE, FALSE)
-- synthesis translate_on
port map (CLK0 => user_CLK0,
    CLK180 => user_CLK180,
    CLK270 => user_CLK270,
    CLK2X => user_CLK2X,
    CLK2X180 => user_CLK2X180,
    CLK90 => user_CLK90,
    CLKDV => user_CLKDV,
    LOCKED => user_LOCKED,
    CLKFB => user_CLKFB,
    CLKIN0 => user_CLKIN,
    RST => user_RST);

Verilog Instantiation Template

CLKDLLE CLKDLLE_instance_name (.CLK0 (user_CLK0),
    .CLK180 (user_CLK180),
    .CLK270 (user_CLK270),
    .CLK2X (user_CLK2X),
    .CLK2X180 (user_CLK2X180),
    .CLK90 (user_CLK90),
    .CLKDV (user_CLKDV),
    .LOCKED (user_LOCKED),
    .CLKFB (user_CLKFB),
    .CLKIN (user_CLKIN),
    .RST (user_RST));

defparam CLKDLLE_instance_name.CLKDV_DIVIDE = integer_value;
// 1.5,2,2.5,3,4,5,8,16 are valid for CLKDV_DIVIDE
defparam CLKDLLE_instance_name.DUTY_CYCLE_CORRECTION = boolean_value; // (TRUE, FALSE)
defparam CLKDLLE_instance_name.STARTUP_WAIT = boolean_value; // (TRUE, FALSE)
**Note:** Additional syntax may be necessary in order to pass the CLKDFFE attributes via the synthesis tool. The above defparam statements may need to be isolated from the synthesis tool with translate_off/translate_on directives. See your synthesis tool documentation for more information on Verilog attribute passing to ensure that you properly pass these attributes to the synthesis tool. Otherwise, you may pass these attributes to the UCF file.

Commonly Used Constraints

STARTUP_WAIT, DUTY_CYCLE_CORRECTION, CLKDV_DIVIDE, and LOC
CLKDLLHF

High Frequency Clock Delay Locked Loop

Architectures Supported

<table>
<thead>
<tr>
<th>CLKDLLHF</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Use CLKDLLHF for the Virtex-E DLL in HF mode. In LF mode, both the separate CLKDLLE and CLKDLL primitive can be used.

CLKDLLHF is a high frequency clock delay locked loop used to minimize clock skew. CLKDLLHF synchronizes the clock signal at the feedback clock input (CLKFB) to the clock signal at the input clock (CLKIN). The locked output (LOCKED) is high when the two signals are in phase. The signals are considered to be in phase when their rising edges are within a specific range of each other (see *The Programmable Logic Data Book* for the most current value).

The frequency of the clock signal at the CLKIN input must be in a specific range depending on speed grade (see *The Programmable Logic Data Book* for the most current values). The CLKIN pin must be driven by an IBUFG or a BUFG.

On-chip synchronization is achieved by connecting the CLKFB input to a point on the global clock network driven by a BUFG, a global clock buffer. The BUFG input can only be connected to the CLK0 output of CLKDLLHF. The BUFG connected to the CLKFB input of the CLKDLLHF must be sourced from the CLK0 output of the same CLKDLLHF. The CLKIN input should be connected to the output of an IBUFG, with the IBUFG input connected to a pad driven by the system clock.

Off-chip synchronization is achieved by connecting the CLKFB input to the output of an IBUF, with the IBUF input connected to a pad. Only the CLK0 output can be used. CLK0 must be connected to the input of OBUF, an output buffer.

The duty cycle of the CLK0 output is 50-50 unless the DUTY_CYCLE_CORRECTION attribute is set to FALSE, in which case the duty cycle is the same as that of the CLKIN input. The duty cycle of the phase shifted output (CLK180) is the same as that of the CLK0 output. The frequency of the CLKDV output is determined by the value assigned to the CLKDV_DIVIDE attribute.

The master reset input (RST) resets CLKDLLHF to its initial (power-on) state. The signal at the RST input is asynchronous and must be held High for just 2ns.
CLKDLLHF Outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK0</td>
<td>Clock at 1x CLKin frequency</td>
</tr>
<tr>
<td>CLK180</td>
<td>Clock at 1x CLKin frequency, shifted 180° with regards to CLK0</td>
</tr>
<tr>
<td>CLKDV</td>
<td>Clock at (1/n)x CLKin frequency, n=CLKDV_DIVIDE value. CLKDV is in phase with CLK0.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>CLKDLLHF locked</td>
</tr>
</tbody>
</table>

**Note:** See the "PERIOD Specifications on CLKDLLs and DCM" section of the "Xilinx Constraints P" chapter in the Constraints Guide for additional information on using the TNM, TNM_NET, and PERIOD attributes with CLKDLLHF components.

**Usage**

This component is generally instantiated in the code as it cannot be easily inferred in synthesis tools. Some synthesis tools may allow inference via an attribute. See your synthesis tool documentation. Generally, global buffers (IBUF, BUFG) are instantiated with the CLKDLLHF component to construct the proper clocking circuit. See the XAPP 132 application note, "Using the Virtex Delay-Locked Loop" and the Xilinx Databook for more information on using the CLKDLLHF component.

**VHDL Instantiation Template**

-- Component Declaration for CLKDLLHF should be placed
-- after architecture statement but before begin keyword

```
component CLKDLLHF
    -- synthesis translate_off
    generic (CLKDV_DIVIDE : real := 2.0; -- (1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 8.0, 16.0)
             DUTY_CYCLE_CORRECTION : Boolean := TRUE; -- (TRUE, FALSE)
             STARTUP_WAIT : boolean := FALSE); -- (TRUE, FALSE)
    -- synthesis translate_on
    port (CLK0 : out STD_ULOGIC;
          CLK180 : out STD_ULOGIC;
          CLKDV : out STD_ULOGIC;
          LOCKED : out STD_ULOGIC;
          CLKFB : in STD_ULOGIC;
          CLKin : in STD_ULOGIC;
          RST : in STD_ULOGIC);
end component;

-- Component Attribute specification for CLKDLLHF
```
-- should be placed after architecture declaration but
-- before the begin keyword

generic map(CLKDV_DIVIDE => real_value, -- (1.5,2,2.5,3,4,5,8,16)
            DUTY_CYCLE_CORRECTION => boolean_value, -- (TRUE, FALSE)
            STARTUP_WAIT => boolean_value); -- (TRUE, FALSE)
.CLK180 (user_CLK180),
.CLKD (user_CLKDV),
.LOCKED (user_LOCKED),
.CLKFB (user_CLKFB),
.CLKIN (user_CLKIN),
.RST (user_RST));

defparam CLKDLLHF_instance_name.CLKD_DIVIDE = integer_value;
// 1.5,2,2.5,3,4,5,8,16 are valid for CLKDV_DIVIDE

defparam CLKDLLHF_instance_name.DUTY_CYCLE_CORRECTION = boolean_value; // (TRUE,FALSE)

defparam CLKDLLHF_instance_name.STARTUP_WAIT = boolean_value; // (TRUE, FALSE)

Note: Additional syntax may be necessary in order to pass the CLKDLLHF attributes via the synthesis tool. The above defparam statements may need to be isolated from the synthesis tool with translate_off/translate_on directives. See your synthesis tool documentation for more information on Verilog attribute passing to ensure that you properly pass these attributes to the synthesis tool. Otherwise, you may pass these attributes to the UCF file.

Commonly Used Constraints
STARTUP_WAIT
DUTY_CYCLE_CORRECTION
CLKDV_DIVIDE
LOC
COMP2, 4, 8, 16

2-, 4-, 8-, 16-Bit Identity Comparators

Architectures Supported

<table>
<thead>
<tr>
<th>COMP2, COMP4, COMP8, COMP16</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

COMP2, COMP4, COMP8, and COMP16 are, respectively, 2-, 4-, 8-, and 16-bit identity comparators. The equal output (EQ) of the COMP2 2-bit, identity comparator is High when the two words A1 – A0 and B1 – B0 are equal. EQ is high for COMP4 when A3 – A0 and B3 – B0 are equal; for COMP8, when A7 – A0 and B7 – B0 are equal; and for COMP16, when A15 – A0 and B15 – B0 are equal.

Equality is determined by a bit comparison of the two words. When any two of the corresponding bits from each word are not the same, the EQ output is Low.
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture behavioral of comp2 is

begin
  process (A, B)
  begin
    if (A=B) then
      EQ <= '1';
    else
      EQ <= '0';
    end if;
  end process;
end behavioral;
Verilog Inference Code

```verilog
always @ (A or B)
  begin
    if (A == B)
      EQ <= 1;
    else
      EQ <= 0;
  end
```
COMPM2, 4, 8, 16

2-, 4-, 8-, 16-Bit Magnitude Comparators

Architectures Supported

<table>
<thead>
<tr>
<th>COMPM2, COMPM4, COMPM8, COMPM16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

COMPM2, COMPM4, COMPM8, and COMPM16 are, respectively, 2-, 4-, 8-, and 16-bit magnitude comparators that compare two positive binary-weighted words.

COMPM2 compares A1 – A0 and B1 – B0, where A1 and B1 are the most significant bits. COMPM4 compares A3 – A0 and B3 – B0, where A3 and B3 are the most significant bits. COMPM8 compares A7 – A0 and B7 – B0, where A7 and B7 are the most significant bits. COMPM16 compares A15 – A0 and B15 – B0, where A15 and B15 are the most significant bits.

The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be measured with this macro by comparing both outputs with a NOR gate.

COMPM2 Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
## COMPM4 Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3, B3</td>
<td>A2, B2</td>
</tr>
<tr>
<td>A3&gt;B3</td>
<td>X</td>
</tr>
<tr>
<td>A3&lt;B3</td>
<td>X</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2&gt;B2</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2&lt;B2</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2=B2</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2=B2</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2=A2</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2=B2</td>
</tr>
<tr>
<td>A3=B3</td>
<td>A2=B2</td>
</tr>
</tbody>
</table>

## COMPM8 Truth Table (also representative of COMPM16)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A7, B7</td>
<td>A6, B6</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>X</td>
</tr>
<tr>
<td>A7&lt;B7</td>
<td>X</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6&gt;B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6&lt;B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>A6&gt;B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>A6&gt;B6</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>A6&gt;B6</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>A6&gt;B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>A6&gt;B6</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
</tbody>
</table>
COMPM8 Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
COMPM8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, these design elements are supported for inference rather than instantiation.

VHDL Inference Code

architecture Behavioral of compm2 is

begin
    process (A,B)
    begin
        if (A>B) then
            GT <= '1';
            LT <= '0';
        elsif (A<B) then
            GT <= '0';
            LT <= '1';
        else
            GT <= '0';
            LT <= '0';
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @(A or B)

    begin
        if (A > B)
        begin
            GT <= 1;
            LT <= 0;
        end
        else if (A < B)
        begin
            GT <= 0;
            LT <= 1;
        end
        else
        begin
            GT <= 0;
            LT <= 0;
        end
    end
### COMP MC8, 16

**8-, 16-Bit Magnitude Comparators**

**Architectures Supported**

<table>
<thead>
<tr>
<th>COMP MC, COMP MC8, COMP MC16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro,</td>
</tr>
<tr>
<td>Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

COMP MC8 is an 8-bit, magnitude comparator that compares two positive binary-weighted words A7 – A0 and B7 – B0, where A7 and B7 are the most significant bits. COMP MC16 is a 16-bit, magnitude comparator that compares two positive binary-weighted words A15 – A0 and B15 – B0, where A15 and B15 are the most significant bits.

These comparators are implemented using carry logic with relative location constraints to ensure efficient logic placement.

The greater-than output (GT) is High when A>B, and the less-than output (LT) is High when A<B. When the two words are equal, both GT and LT are Low. Equality can be flagged with this macro by connecting both outputs to a NOR gate.

### COMP MC8 Truth Table (also representative of COMP MC16)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A7, B7</td>
<td>A6, B6</td>
</tr>
<tr>
<td>A5, B5</td>
<td>A4, B4</td>
</tr>
<tr>
<td>A3, B3</td>
<td>A2, B2</td>
</tr>
<tr>
<td>A1, B1</td>
<td>A0, B0</td>
</tr>
<tr>
<td></td>
<td>GT</td>
</tr>
<tr>
<td>A7&gt;B7</td>
<td>1</td>
</tr>
<tr>
<td>A7&lt;B7</td>
<td>0</td>
</tr>
<tr>
<td>A7=B7</td>
<td>1</td>
</tr>
<tr>
<td>A6&gt;B6</td>
<td>0</td>
</tr>
<tr>
<td>A6&lt;B6</td>
<td>0</td>
</tr>
<tr>
<td>A6=B6</td>
<td>0</td>
</tr>
<tr>
<td>A5&gt;B5</td>
<td>0</td>
</tr>
<tr>
<td>A5=B5</td>
<td>0</td>
</tr>
<tr>
<td>A4&gt;B4</td>
<td>0</td>
</tr>
<tr>
<td>A4&lt;B4</td>
<td>0</td>
</tr>
<tr>
<td>A3&gt;B3</td>
<td>0</td>
</tr>
<tr>
<td>A3=B3</td>
<td>0</td>
</tr>
<tr>
<td>A2&gt;B2</td>
<td>0</td>
</tr>
<tr>
<td>A2=B2</td>
<td>0</td>
</tr>
<tr>
<td>A1&gt;B1</td>
<td>0</td>
</tr>
<tr>
<td>A1=B1</td>
<td>0</td>
</tr>
</tbody>
</table>
COMPMC8 Truth Table (also representative of COMPMC16)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A7, B7</td>
<td>A6, B6</td>
</tr>
<tr>
<td>A5, B5</td>
<td>A4, B4</td>
</tr>
<tr>
<td>A3, B3</td>
<td>A2, B2</td>
</tr>
<tr>
<td>A1, B1</td>
<td>A0, B0</td>
</tr>
<tr>
<td>GT</td>
<td>LT</td>
</tr>
<tr>
<td>A7=B7</td>
<td>A6=B6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

COMPMC8 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
COMPMC8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

For HDL, these design elements are supported for inference rather than instantiation.

**VHDL Inference Code**

```vhdl
architecture Behavioral of compmc8 is
begin
    process (A,B)
    begin
        if (A>B) then
            GT <= '1';
            LT <= '0';
        elsif (A<B) then
            GT <= '0';
            LT <= '1';
        else
            GT <= '0';
            LT <= '0';
        end if;
    end process;
end Behavioral;
```

**Verilog Inference Code**

```verilog
always @(A or B)
begin
    if (A > B)
        begin
            GT <= 1;
            LT <= 0;
        end
    else if (A < B)
        begin
            GT <= 0;
            LT <= 1;
        end
    else
        begin
            GT <= 0;
            LT <= 0;
        end
end
```
CR8CE, CR16CE

8-, 16-Bit Negative-Edge Binary Ripple Counters with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>CR8CE, CR16CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

CR8CE and CR16CE are 8-bit and 16-bit, cascadable, clearable, binary, ripple counters. The asynchronous clear (CLR), when High, overrides all other inputs and causes the Q outputs to go to logic level zero. The counter increments when the clock enable input (CE) is High during the High-to-Low clock (C) transition. The counter ignores clock transitions when CE is Low.

Larger counters can be created by connecting the last Q output (Q7 for CR8CE, Q15 for CR16CE) of the first stage to the clock input of the next stage. CLR and CE inputs are connected in parallel. The clock period is not affected by the overall length of a ripple counter. The overall clock-to-output propagation is \( n(t_{C \rightarrow Q}) \), where \( n \) is the number of stages and the time \( t_{C \rightarrow Q} \) is the C-to-Qz propagation delay of each stage.

The counter is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( z = 7 \) for CR8CE; \( z = 15 \) for CR16CE.
CR8CE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```
architecture Behavioral of cr8ce is
begin
```
process (C, CLR)
begin
    if (CLR='1') then
        Q <= (others => '0');
    elseif (C' event and C='0') then
        if (CE='1') then
            Q <= Q + 1;
        end if;
    end if;
end process;

end Behavioral;

Verilog Inference Code
always @(posedge C or posedge CLR)
begin
    if (CLR)
        Q <= 0;
    else if (CE)
        Q <= Q + 1;
end
CRD8CE, CRD16CE

8-, 16-Bit Dual-Edge Triggered Binary Ripple Counters with Clock Enable and Asynchronous Clear

Architectures Supported

| CRD8CE, CRD16CE       | Spartan-II, Spartan-IIIE | No          
|-----------------------|--------------------------|-------------
|                       | Spartan-3                | No          
|                       | Virtex, Virtex-E         | No          
|                       | Virtex-II, Virtex-II Pro, Virtex-II Pro X | No 
|                       | XC9500, XC9500XV, XC9500XL | No          
| CoolRunner XPLA3      | No                       
| CoolRunner-II         | Macro                    

CRD8CE and CRD16CE are dual edge triggered 8-bit and 16-bit, cascadable, clearable, binary, ripple counters. The asynchronous clear (CLR), when High, overrides all other inputs and causes the Q outputs to go to logic level zero. The counter increments when the clock enable input (CE) is High during the High-to-Low and Low-to-High clock (C) transitions. The counter ignores clock transitions when CE is Low.

Larger counters can be created by connecting the last Q output (Q7 for CRD8CE, Q15 for CRD16CE) of the first stage to the clock input of the next stage. CLR and CE inputs are connected in parallel. The clock period is not affected by the overall length of a ripple counter. The overall clock-to-output propagation is \( n(t_{C\rightarrow Q}) \), where \( n \) is the number of stages and the time \( t_{C\rightarrow Q} \) is the C-to-Qz propagation delay of each stage.

The counter is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( z = 7 \) for CR8CE; \( z = 15 \) for CR16CE.
CRD8CE Implementation CoolRunner-II
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of crd8ce is
begin
    process (C, CLR)
    begin
        if (CLR='1') then
            Q <= (others => '0');
        elsif (C' event) then
            if (CE='1') then
                Q <= Q + 1;
            end if;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C or posedge CLR)
begin
    if (CLR)
        Q <= 0;
    else if (CE)
        Q <= Q + 1;
end
**D2_4E**

2- to 4-Line Decoder/Demultiplexer with Enable

**Architectures Supported**

<table>
<thead>
<tr>
<th>D2_4E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

When the enable (E) input of the D2_4E decoder/demultiplexer is High, one of four active-High outputs (D3 – D0) is selected with a 2-bit binary address (A1 – A0) input. The non-selected outputs are Low. Also, when the E input is Low, all outputs are Low. In demultiplexer applications, the E input is the data input.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>A0</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Usage**

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of d2_4e is

begin

process (A, E)
begin
if (E='0') then
D <= "0000";
else
case A is
when "00" =>
D <= "0001";
when "01" =>
D <= "0010";
when "10" =>
D <= "0100";
when "11" =>
D <= "1000";
when others =>
D <= "0000";
end case;
end if;
end process;

end Behavioral;

Verilog Inference Code

always @(A or E)
begin
if (!E)
D <= 4'b0000;
else
begin
case (A)
2'b00 : D <= 4'b0001;
2'b01 : D <= 4'b0010;
2'b10 : D <= 4'b0100;
2'b11 : D <= 4'b1000;
endcase
end
end
D3_8E

3- to 8-Line Decoder/Demultiplexer with Enable

Architectures Supported

<table>
<thead>
<tr>
<th>D3_8E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

When the enable (E) input of the D3_8E decoder/demultiplexer is High, one of eight active-High outputs (D7 – D0) is selected with a 3-bit binary address (A2 – A0) input. The non-selected outputs are Low. Also, when the E input is Low, all outputs are Low. In demultiplexer applications, the E input is the data input.

<table>
<thead>
<tr>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of d3_8e is

begin

process (A, E)
begin
  if (E='0') then
    D <= "00000000";
  else
    case A is
      when "000" =>
        D <= "00000001";
      when "001" =>
        D <= "00000010";
      when "010" =>
        D <= "00000100";
      when "011" =>
        D <= "00001000";
      when others =>
        D <= "00000000";
    end case;
  end if;
end process;

end Behavioral;
when "100" =>
    D <= "00010000";
when "101" =>
    D <= "00100000";
when "110" =>
    D <= "01000000";
when "111" =>
    D <= "10000000";
when others =>
    D <= "00000000";
end case;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (A or E)
begin
    if (!E)
        D <= 8'b00000000;
    else
        begin
            case (A)
                3'b000 : D <= 8'b00000001;
                3'b001 : D <= 8'b00000010;
                3'b010 : D <= 8'b00000100;
                3'b011 : D <= 8'b00001000;
                3'b100 : D <= 8'b00010000;
                3'b101 : D <= 8'b00100000;
                3'b110 : D <= 8'b01000000;
                3'b111 : D <= 8'b10000000;
            endcase
        end
    end
end
D4_16E

4- to 16-Line Decoder/Demultiplexer with Enable

Architectures Supported

<table>
<thead>
<tr>
<th>D4_16E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

When the enable (E) input of the D4_16E decoder/demultiplexer is High, one of 16 active-High outputs (D15 – D0) is selected with a 4-bit binary address (A3 – A0) input. The non-selected outputs are Low. Also, when the E input is Low, all outputs are Low. In demultiplexer applications, the E input is the data input.

See “D3_8E” for a representative truth table derivation.

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of d4_16e is
begin
process (A, E)
begin
```
if (E='0') then
  D <= "0000000000000000";
else
  case A is
    when "0000" =>
      D <= "0000000000000001";
    when "0001" =>
      D <= "0000000000000010";
    when "0010" =>
      D <= "0000000000000100";
    when "0011" =>
      D <= "0000000000001000";
    when "0100" =>
      D <= "0000000000010000";
    when "0101" =>
      D <= "0000000000100000";
    when "0110" =>
      D <= "0000000001000000";
    when "0111" =>
      D <= "0000000010000000";
    when "1000" =>
      D <= "0000000100000000";
    when "1001" =>
      D <= "0000001000000000";
    when "1010" =>
      D <= "0000010000000000";
    when "1011" =>
      D <= "0000100000000000";
    when "1100" =>
      D <= "0001000000000000";
    when "1101" =>
      D <= "0010000000000000";
    when "1110" =>
      D <= "0100000000000000";
    when "1111" =>
      D <= "1000000000000000";
    when others =>
      D <= "0000000000000000";
  end case;
end if;
end process;

end Behavioral;

Verilog Inference Code

always @ (A or E)
begin
  if (!E)
    D <= 16'b0000000000000000;
  else
    begin
      case (A)
        4'b0000 : D <= 16'b0000000000000001;
        4'b0001 : D <= 16'b0000000000000010;
        4'b0010 : D <= 16'b0000000000000100;
        4'b0011 : D <= 16'b0000000000001000;
        4'b0100 : D <= 16'b0000000000010000;
        4'b0101 : D <= 16'b0000000000100000;
        4'b0110 : D <= 16'b0000000001000000;
        4'b0111 : D <= 16'b0000000010000000;
        4'b1000 : D <= 16'b0000000100000000;
        4'b1001 : D <= 16'b0000001000000000;
        4'b1010 : D <= 16'b0000010000000000;
        4'b1011 : D <= 16'b0000100000000000;
        4'b1100 : D <= 16'b0001000000000000;
        4'b1101 : D <= 16'b0010000000000000;
        4'b1110 : D <= 16'b0100000000000000;
        4'b1111 : D <= 16'b1000000000000000;
      end case;
    end
  end
end
4'b0101 : D <= 16'b0000000000100000;
4'b0110 : D <= 16'b0000000001000000;
4'b0111 : D <= 16'b0000000010000000;
4'b1000 : D <= 16'b0000000100000000;
4'b1001 : D <= 16'b0000001000000000;
4'b1010 : D <= 16'b0000010000000000;
4'b1011 : D <= 16'b0000100000000000;
4'b1100 : D <= 16'b0001000000000000;
4'b1101 : D <= 16'b0010000000000000;
4'b1110 : D <= 16'b0100000000000000;
4'b1111 : D <= 16'b1000000000000000;
endcase
end
end
DCM

Digital Clock Manager

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>DCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-III</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

DCM is a digital clock manager that provides multiple functions. It can implement a clock delay locked loop, a digital frequency synthesizer, digital phase shifter, and a digital spread spectrum.

**Note:** All unused inputs must be driven Low. The program will automatically tie the inputs Low if they are unused.

Clock Delay Locked Loop (DLL)

DCM includes a clock delay locked loop used to minimize clock skew for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X devices. DCM synchronizes the clock signal at the feedback clock input (CLKFB) to the clock signal at the input clock (CLKIN). The locked output (LOCKED) is high when the two signals are in phase. The signals are considered to be in phase when their rising edges are within a specified time (ps) of each other. See *The Programmable Logic Data Book* for the specified time value.

DCM supports two frequency modes for the DLL. By default, the DLL_FREQUENCY_MODE attribute is set to Low and the frequency of the clock signal at the CLkin input must be in the Low (DLL_CLKIN_MIN_LF to DLL_CLKIN_MAX_LF) frequency range (MHz). See *The Programmable Logic Data Book* for the current DLL_CLKIN_MIN_LF to DLL_CLKIN_MAX_LF frequency range values. In Low frequency mode, the CLK0, CLK90, CLK180, CLK270, CLK2X, CLKDV, and CLK2X180 outputs are available.

When the DLL_FREQUENCY_MODE attribute is set to High, the frequency of the clock signal at the CLkin input must be in the High (DLL_CLKIN_MIN_HF to DLL_CLKIN_MAX_HF) frequency range (MHz). See *The Programmable Logic Data Book* for the current DLL_CLKIN_MIN_HF to DLL_CLKIN_MAX_HF frequency range values. In High frequency mode, only the CLK0, CLK180, and CLKDV outputs are available.

On-chip synchronization is achieved by connecting the CLKFB input to a point on the global clock network driven by a BUFG, a global clock buffer. The BUFG connected to the CLKFB input of the DCM must be sourced from either the CLK0 or CLK2X outputs of the same DCM. The CLkin input should be connected to the output of an IBUF, with the IBUF input connected to a pad driven by the system clock.

Off-chip synchronization is achieved by connecting the CLKFB input to the output of an IBUF, with the IBUF input connected to a pad. Either the CLK0 or CLK2X...
output can be used but not both. The CLK0 or CLK2X must be connected to the input of OBUF, an output buffer. The CLK_FEEDBACK attribute controls whether the CLK0 output, the default, or the CLK2X output is the source of the CLKFB input.

The duty cycle of the CLK0 output is 50-50 unless the DUTY_CYCLE_CORRECTION attribute is set to FALSE, in which case the duty cycle is the same as that of the CLKin input. The duty cycle of the phase shifted outputs (CLK90, CLK180, and CLK270) is the same as that of the CLK0 output. The duty cycle of the CLK2X, CLK2X180, and CLKDV outputs is 50-50 unless CLKDV_DIVIDE is a non-integer and the DLL_FREQUENCY_MODE is High (see “CLKDV_DIVIDE,” in the Constraints Guide for details). The frequency of the CLKDV output is determined by the value assigned to the CLKDV_DIVIDE attribute.

### DCM Clock Delay Lock Loop Outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK0</td>
<td>Clock at 1x CLKin frequency</td>
</tr>
<tr>
<td>CLK180</td>
<td>Clock at 1x CLK0 frequency, shifted 180° with regards to CLK0</td>
</tr>
<tr>
<td>CLK270*</td>
<td>Clock at 1x CLK0 frequency, shifted 270° with regards to CLK0</td>
</tr>
<tr>
<td>CLK2X*</td>
<td>Clock at 2x CLK0 frequency, in phase with CLK0</td>
</tr>
<tr>
<td>CLK2X180*</td>
<td>Clock at 2x CLK0 frequency shifted 180° with regards to CLK2X</td>
</tr>
<tr>
<td>CLK90*</td>
<td>Clock at 1x CLK0 frequency, shifted 90° with regards to CLK0</td>
</tr>
<tr>
<td>CLKDV</td>
<td>Clock at (1/n) x CLK0 frequency, where n=CLKDV_DIVIDE value. CLKDV is in phase with CLK0.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>All enabled DCM features locked.</td>
</tr>
</tbody>
</table>

* The CLK90, CLK270, CLK2X, and CLK2X180 outputs are not available if the DLL_FREQUENCY_MODE is set to High.

### Digital Frequency Synthesizer (DFS)

The CLKFX and CLKFX180 outputs in conjunction with the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes provide a frequency synthesizer that can be any multiple or division of CLKin. CLKFX and CLKin are in phase every CLKFX_MULTIPLY cycles of CLKFX and every CLKFX_DIVIDE cycles of CLKin when a feedback is provided to the CLKFB input of the DLL. The frequency of CLKFX is defined by the following equation.

\[
\text{Frequency}_{CLKFX} = \left( \frac{\text{CLKFX_MULTIPLY_value}}{\text{CLKFX_DIVIDE_value}} \right) \times \text{Frequency}_{CLKin}
\]

Both the CLKFX or CLKFX180 output can be used simultaneously.

CLKFX180 is 1x the CLKFX frequency, shifted 180° with regards to CLKFX. CLKFX and CLKFX180 always have a 50/50 duty cycle.

The DFS_FREQUENCY_MODE attribute specifies the allowable input clock and output clock frequency ranges.

The CLK_FEEDBACK attribute set to NONE will cause the DCM to be in the Digital Frequency Synthesizer mode. The CLKFX and CLKFX180 will be generated without phase correction with respect to CLKin.

See The Programmable Logic Data Book for the allowable frequency range of CLKFX.
Digital Phase Shifter (DPS)

The phase shift (skew) between the rising edges of CLKIN and CLKFB may be configured as a fraction of the CLKIN period with the PHASE_SHIFT attribute. This allows the phase shift to remain constant as ambient conditions change. The CLKOUT_PHASE_SHIFT attribute controls the use of the PHASE_SHIFT value. By default, the CLKOUT_PHASE_SHIFT attribute is set to NONE and the PHASE_SHIFT attribute has no effect.

By creating skew between CLKIN and CLKFB, all DCM output clocks are phase shifted by the amount of the skew.

When the CLKOUT_PHASE_SHIFT attribute is set to FIXED, the skew set by the PHASE_SHIFT attribute is used at configuration for the rising edges of CLKIN and CLKFB. The skew remains constant.

When the CLKOUT_PHASE_SHIFT attribute is set to VARIABLE, the skew set at configuration is used as a starting point and the skew value can be changed dynamically during operation using the PS* signals. This digital phase shifter feature is controlled by a synchronous interface. The inputs PSEN (phase shift enable) and PSINCDEC (phase shift increment/decrement) are set up to the rising edge of PSCLK (phase shift clock). The PSDONE (phase shift done) output is clocked with the rising edge of PSCLK (the phase shift clock). PSDONE must be connected to implement the complete synchronous interface. The rising-edge skew between CLKIN and CLKFB may be dynamically adjusted after the LOCKED output goes High.

The PHASE_SHIFT attribute value specifies the initial phase shift amount when the device is configured. Then the PHASE_SHIFT value is changed one unit when PSEN is activated for one period of PSCLK. The PHASE_SHIFT value is incremented when PSINCDEC is High and decremented when PSINCDEC is Low during the period that PSEN is High. When the DCM completes an increment or decrement operation, the PSDONE output goes High for a single PSCLK cycle to indicate the operation is complete. At this point the next change may be made. When RST (reset) is High, the PHASE_SHIFT attribute value is reset to the skew value set at configuration.

If CLKOUT_PHASE_SHIFT is FIXED or NONE, the PSEN, PSINCDEC, and PSCLK inputs must be tied to GND. The program will automatically tie the inputs to GND if they are not connected by the user.

Digital Spread Spectrum (DSS)

The digital spread spectrum function of DCM broadens the frequency spectrum of the output clocks to help meet FCC requirements. Spread spectrum clocking is a simple method to lower electromagnetic interference (EMI). A digital system can create a severe EMI spike at the clock frequency. Spread spectrum clocking speeds up and slows down the clock within a few percent of the target frequency, thus flattening out the EMI peak by spreading it across a range of frequencies.

When the DSS_MODE attribute is set to NONE, the DSSEN input has no effect. When DSS_MODE is set to a value other than NONE and DSSEN is High, the output clock frequency is modulated by the amount of spread specified by the DSS_MODE value.

For Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the DSS feature of the DCM has limited EMI reduction capability.

For maximum EMI reduction an IP core available from Xilinx is recommended.

**Note:** DSS is not supported in Spartan-3.
Additional Status Bits

The STATUS output bits return the following information.

**DCM Additional Status Bits**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0   | Phase Shift Overflow*  
     | $1 \cdot |\text{PHASE\_SHIFT}| > 255$ |
| 1   | DLL CLKIN stopped**  
     | $1 \cdot \text{CLKIN stopped toggling}$ |
| 2   | DLL CLKFX stopped  
     | $1 \cdot \text{CLKFX stopped toggling}$ |
| 3   | No |
| 4   | No |
| 5   | No |
| 6   | No |
| 7   | No |

* Phase Shift Overflow will also go high if the end of the phase shift delay line is reached (see the product data sheet for the most current value of the maximum shifting delay).

** If only the DFS outputs are used (CLKFX & CLKFX180), this status bit will not go high if CLKIN stops.

**LOCKED**

When LOCKED is high, all enabled signals are locked.

**RST**

The master reset input (RST) resets DCM to its initial (power-on) state. The signal at the RST input is asynchronous and must be held High for 2ns.

**Usage**

This component is generally instantiated in the code as it cannot be easily inferred in synthesis tools. Some synthesis tools may allow inference via an attribute. See your synthesis tool documentation.

**VHDL Instantiation Template**

-- Component Declaration for DCM should be placed  
-- after architecture statement but before begin keyword

component DCM  
  -- synthesis translate_off
  generic (CLK\_FEEDBACK : string := "1X";
  CLKDV\_DIVIDE : real := 2.0; -- (1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 8.0, 16.0)  
  CLKFX\_DIVIDE : integer := 1; -- (1 to 4096)
CLKFX_MULTIPLY : integer := 4; -- (1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 5.5, 
-- 6.0, 6.5, 7.0, 7.5, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0)
CLKin_DIVIDE_BY_2 : boolean := FALSE; -- (TRUE, FALSE)
CLKOUT_PHASE_SHIFT : string := "NONE";
DESEKW_ADJUST : string := "SYSTEM_SYNCHRONOUS";
DFS_FREQUENCY_MODE : string := "LOW";
DLL_FREQUENCY_MODE : string := "LOW";
DSS_MODE : string := "NONE";
DUTY_CYCLE_CORRECTION : Boolean := TRUE; -- (TRUE, FALSE)
PHASE_SHIFT : real := 0;
STARTUP_WAIT : boolean := FALSE); -- (TRUE, FALSE)
-- synthesis translate_on
port (CLK0 : out STD_ULOGIC;
    CLK180 : out STD_ULOGIC;
    CLK270 : out STD_ULOGIC;
    CLK2X : out STD_ULOGIC;
    CLK2X180 : out STD_ULOGIC;
    CLK90 : out STD_ULOGIC;
    CLKDV : out STD_ULOGIC;
    CLKFX : out STD_ULOGIC;
    CLKFX180 : out STD_ULOGIC;
    LOCKED : out STD_ULOGIC;
    PSDONE : out STD_ULOGIC;
    STATUS : out STD_LOGIC_VECTOR (7 downto 0);
    CLKFB : in STD_ULOGIC;
    CLKin : in STD_ULOGIC;
    DSSEN : in STD_ULOGIC;
    PSCLK : in STD_ULOGIC;
    PSEN : in STD_ULOGIC;
    PSINCDEC : in STD_ULOGIC;
    RST : in STD_ULOGIC);
end component;

-- Component Attribute specification for DCM
-- should be placed after architecture declaration but
attribute CLK_FEEDBACK : string;
attribute CLKDV_DIVIDE : real;
attribute CLKFX_DIVIDE : integer;
attribute CLKFX_MULTIPLY : integer;
attribute CLKIN_DIVIDE_BY_2 : string;
attribute CLKOUT_PHASE_SHIFT : string;
attribute DESKEW_ADJUST : string;
attribute DFS_FREQUENCY_MODE : string;
attribute DLL_FREQUENCY_MODE : string;
attribute DSS_MODE : string;
attribute DUTY_CYCLE_CORRECTION : string;
attribute PHASE_SHIFT : real;
attribute STARTUP_WAIT : boolean;

attribute CLK_FEEDBACK of DCM_instance_name: label is "1X";
attribute CLKDV_DIVIDE of DCM_instance_name: label is 2.0;
-- (1.5,2,2.5,3,4, 5, 8, 16) are valid for CLKDV_DIVIDE
attribute CLKFX_DIVIDE of DCM_instance_name: label is 1;
attribute CLKFX_MULTIPLY of DCM_instance_name: label is 4;
attribute CLKIN_DIVIDE_BY_2 of DCM_instance_name: label is "FALSE";
attribute CLKOUT_PHASE_SHIFT of DCM_instance_name: label is "NONE";
attribute DESKEW_ADJUST of DCM_instance_name: label is
"SYSTEM_SYNCHRONOUS";
attribute DFS_FREQUENCY_MODE of DCM_instance_name: label is "LOW";
attribute DLL_FREQUENCY_MODE of DCM_instance_name: label is "LOW";
attribute DSS_MODE of DCM_instance_name: label is "NONE";
attribute DUTY_CYCLE_CORRECTION of DCM_instance_name: label is TRUE;
-- (TRUE, FALSE) are valid for DUTY_CYCLE_CORRECTION
attribute PHASE_SHIFT of DCM_instance_name: label is 0;
attribute STARTUP_WAIT of DCM_instance_name: label is FALSE; -- (TRUE,FALSE)

-- Component Instantiation for DCM should be placed
DCM_INSTANCE_NAME : DCM

-- in architecture after the begin keyword

DCM_INSTANCE_NAME : DCM

-- synthesis translate_off

generic map(CLK_FEEDBACK => "string_value",
    CLKDV_DIVIDE => real_value, -- (1.5,2,2.5,3,4,5,8,16)
    CLKFX_DIVIDE => integer_value,
    CLKFX_MULTIPLY => integer_value,
    CLKin_DIVIDE_BY_2 => boolean_value, -- (TRUE, FALSE)
    CLKOUT_PHASE_SHIFT => "string_value",
    DESKEW_ADJUST => "string_value",
    DFS_FREQUENCY_MODE => "string_value",
    DLL_FREQUENCY_MODE => "string_value",
    DSS_MODE => "string_value",
    DUTY_CYCLE_CORRECTION => boolean_value, -- (TRUE, FALSE)
    PHASE_SHIFT => integer_value,
    STARTUP_WAIT => boolean) -- (TRUE, FALSE)

-- synthesis translate_on

port map (CLK0 => userCLK0,
    CLK180 => userCLK180,
    CLK270 => userCLK270,
    CLK2X => userCLK2X,
    CLK2X180 => userCLK2X180,
    CLK90 => userCLK90,
    CLKDV => userCLKDV,
    CLKFX => userCLKFX,
    CLKFX180 => userCLKFX180,
    LOCKED => userLOCKED)

PSDONE => userPSDONE,
STATUS => userSTATUS,
CLKFB => userCLKFB,
CLKin => userCLKin,
DSEN => userDSEN,
PSCLK => userPSCLK,
PSEN => userPSEN,
PSINCDEC => userPSINCDEC,
RST => user_RST);

Verilog Instantiation Template

DCM DCM_instance_name (.CLK0 (user_CLK0),
   .CLK180 (user_CLK180),
   .CLK270 (user_CLK270),
   .CLK2X (user_CLK2X),
   .CLK2X180 (user_CLK2X180),
   .CLK90 (user_CLK90),
   .CLKDV (user_CLKDV),
   .CLKFX (user_CLKFX),
   .CLKFX180 (user_CLKFX180),
   .LOCKED (user_LOCKED),
   .PSDONE (user_PSDONE),
   .STATUS (user_STATUS),
   .CLKFB (user_CLKFB),
   .CLINKIN (user_CLINKIN),
   .DSSEN (user_DSSEN),
   .PSCLK (user_PSClk),
   .PSEN (user_PSEN),
   .PSINCDEC (user_PSINCDEC),
   .RST (user_RST));

defparam DCM_instance_name.CLK_FEEDBACK => "string_value";
defparam DCM_instance_name.CLKDV_DIVIDE = integer_value;
   // (1.5,2,2.5,3,4,5,8,16)
defparam DCM_instance_name.CLKFX_DIVIDE => integer_value;
defparam DCM_instance_name.CLKFX_MULTIPLY => integer_value;
defparam DCM_instance_name.CLKIN_DIVIDE_BY_2 => boolean_value; // (TRUE, FALSE)
defparam DCM_instance_name.CLKOUT_PHASE_SHIFT => "string_value";
defparam DCM_instance_name.DESKEW_ADJUST => "string_value";
defparam DCM_instance_name.DFS_FREQUENCY_MODE => "string_value";
defparam DCM_instance_name.DLL_FREQUENCY_MODE => "string_value";
defparam DCM_instance_name.DSS_MODE => "string_value";
defparam DCM_instance_name.DUTY_CYCLE_CORRECTION => "string_value"; // (TRUE, FALSE)
defparam DCM_instance_name.PHASE_SHIFT => integer_value;
defparam DCM_instance_name.STARTUP_WAIT => boolean_value; // (TRUE, FALSE)

**Note:** Additional syntax may be necessary in order to pass the DCM attributes via the synthesis tool. The above defparam statements may need to be isolated from the synthesis tool with translate_off/translate_on directives. See your synthesis tool documentation for more information on Verilog attribute passing to ensure that you properly pass these attributes to the synthesis tool. Otherwise, you may pass these attributes to the UCF file.

**Commonly Used Constraints**

- CLKDV_DIVIDE
- CLK_FEEDBACK
- CLKFX_DIVIDE
- CLKFX_MULTIPLY
- CLKin_DIVIDE_BY_2
- CLKOUT_PHASE_SHIFT
- DUTY_CYCLE_CORRECTION
- DFS_FREQUENCY_MODE
- DLL_FREQUENCY_MODE
- LOC
- PHASE_SHIFT
- STARTUP_WAIT
- DESKEW_ADJUST
DEC_CC4, DEC_CC8, DEC_CC16

4-, 8-, 16-Bit Active Low Decoders

Architectures Supported

<table>
<thead>
<tr>
<th>DEC_CC4, DEC_CC8, DEC_CC16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

These decoders are used to build wide-decoder functions. They are implemented by cascading CY_MUX elements driven by lookup tables (LUTs). The C_IN pin can only be driven by the output (O) of a previous decode stage. When one or more of the inputs (A) are Low, the output is Low. When all the inputs are High and the C_IN input is High, the output is High. You can decode patterns by adding inverters to inputs.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

z = 3 for DEC_CC4; z = 7 for DEC_CC8; z = 15 for DEC_CC16

The C_IN pin can only be initialized by a CY_INIT or by the output of a previous decode stage.

Usage

DEC_CC4 cannot be directly inferred or instantiated. The proper way to use a DEC_CC4 is to infer the primitive components that make up the DEC_CC4.
DECODE4, 8, 16

4-, 8-, 16-Bit Active-Low Decoders

Architectures Supported

<table>
<thead>
<tr>
<th>DECODE4, DECODE8, DECODE16</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

Inputs Outputs*

<table>
<thead>
<tr>
<th>A0</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 3 for DECODE4, z = 7 for DECODE8, z = 15 for DECODE16

* A pull-up resistor must be connected to the output to establish High-level drive current.
DECODE8 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

Usage

For HDL, these design elements are inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of decode4 is

begin

process (A)
begin
  case A is
    when "1111" => O <= '1';
    when others => O <= '0';
  end case;
end process;

end Behavioral;

Verilog Inference Code

always @(A)
begin
  case (A)
    4'h0 : O <= 1;
    default : O <= 0;
  endcase
end
DECODE32, 64

32- and 64-Bit Active-Low Decoders

Architectures Supported

<table>
<thead>
<tr>
<th>DECODE3, DECODE2, DECODE64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

DECODE32 and DECODE64 are 32- and 64-bit active-low decoders. These decoders are implemented using combinations of LUTs and MUXCYs.

See “DECODE4, 8, 16” for a representative schematic.

<table>
<thead>
<tr>
<th></th>
<th>Inputs</th>
<th></th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>A1</td>
<td>...</td>
<td>Az</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

z = 31 for DECODE32, z = 63 for DECODE64

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of decode32 is
begin
    process (A)
    begin
        case A is
            when x"11111111" => O <= '1';
            when others => O <= '0';
        end case;
    end process;
end Behavioral;
```
Verilog Inference Code

always @(A)
begin
    case (A)
        32'h00000000 : O <= 1;
        default : O <= 0;
    endcase
end
FD

D Flip-Flop

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FD is a single D-type flip-flop with data input (D) and data output (Q). The data on the D inputs is loaded into the flip-flop during the Low-to-High clock (C) transition. The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

See “FD4, 8, 16” for information on multiple D flip-flops for XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

FD Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of fd is
begin
process (C)
begin
if C'event and C='1' then
Q <= D;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C) begin
Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FD should be placed
-- after architecture statement but before begin keyword

component FD
-- synthesis translate_off
-- generic (INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
C : in STD_ULOGIC;
D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FD
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FD_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FD should be placed
-- in architecture after the begin keyword

FD_INSTANCE_NAME : FD
-- synthesis translate_off
-- generic map (INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
C => user_C,
D => user_D);
Verilog Instantiation Template

FD $FD_{\text{instance\_name}}$ (.Q (user_Q),
   .C (user_C),
   .D (user_D));

defparam $FD_{\text{instance\_name}}$.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FD_1

D Flip-Flop with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>FD_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FD_1 is a single D-type flip-flop with data input (D) and data output (Q). The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition. The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fd_1 is
begin
  process (C)
  begin
    if C' event and C = '0' then
      Q <= D;
    end if;
  end process;
end Behavioral;
```
Verilog Inference Code

always @ (negedge C) begin
  Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FD_1 should be placed
-- after architecture statement but before begin keyword

component FD_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FD_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FD_1_instance_name : label is "0";
-- values can be (0 or 1)

begin
  FD_1_INSTANCE_NAME : FD_1
    -- synthesis translate_off
    generic map (INIT => bit_value) -- INIT value can be '0' or '1'
    -- synthesis translate_on
    port map (Q => user_Q,
              C => user_C,
              D => user_D);
end Behavioral

Verilog Instantiation Template

FD_1 FD_1_instance_name (.Q (user_Q),
                         .C (user_C),
                         .D (user_D));

defparam FD_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, and XBLKNM
FD4, 8, 16

Multiple D Flip-Flops

Architectures Supported

<table>
<thead>
<tr>
<th>FD4, FD8, FD16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FD4, FD8, FD16 are multiple D-type flip-flops with data inputs (D) and data outputs (Q). FD4, FD8, and FD16 are, respectively, 4-bit, 8-bit, and 16-bit registers, each with a common clock (C). The data on the D inputs is loaded into the flip-flop during the Low-to-High clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 – D7</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

z = 3 for FD4; z = 7 for FD8; z = 15 for FD16
FD8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fd4 is
begin
    process (C)
    begin
        if C' event and C='1' then
            Q <= D;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C)
begin
    Q <= D;
end
```
FD4CE, FD8CE, FD16CE

4-, 8-, 16-Bit Data Registers with Clock Enable and Asynchronous Clear

**Architectures Supported**

<table>
<thead>
<tr>
<th>FD4CE, FD8CE, FD16CE</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FD4CE, FD8CE, and FD16CE are, respectively, 4-, 8-, and 16-bit data registers with clock enable and asynchronous clear. When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data inputs (D) is transferred to the corresponding data outputs (Q) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data outputs (Q) Low. When CE is Low, clock transitions are ignored.

The flip-flops are asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEx, or STARTUP_VIRTEx2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for FD4CE; z = 7 for FD8CE; z = 15 for FD16CE.

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of fd4ce is
begin
    process (C, CLR) begin
        if (CLR = '1') then
            Q <= "0000";
        elsif (C' event and C = '1') then
            if (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 4'b0000;
  else
    begin
      if (CE)
        Q <= D;
    end
end
FD4RE, FD8RE, FD16RE

4-, 8-, 16-Bit Data Registers with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FD4RE, FD8RE, FD16RE</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FD4RE, FD8RE, and FD16RE are, respectively, 4-, 8-, and 16-bit data registers. When the clock enable (CE) input is High, and the synchronous reset (R) input is Low, the data on the data inputs (D) is transferred to the corresponding data outputs (Q0) during the Low-to-High clock (C) transition. When R is High, it overrides all other inputs and resets the data outputs (Q) Low on the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flops are asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for FD4RE; z = 7 for FD8RE; z = 15 for FD16RE

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fd4re is
begin
    process (C) begin
        if (C' event and C = '1') then
            if (R = '1') then
                Q <= "0000";
            elsif (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```
Verilog Inference Code

always @ (posedge C)
begin
  if (R)
    Q <= 4'b0000;
  else
    begin
      if (CE)
        Q <= D;
    end
end
FDC

D Flip-Flop with Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDC is a single D-type flip-flop with data (D) and asynchronous clear (CLR) inputs and data output (Q). The asynchronous CLR, when High, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop when CLR is Low on the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDC Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of fdc is
begin
    process (C, CLR) begin
        if (CLR = '1') then
            Q <= '0';
        elsif (C' event and C = '1') then
            Q <= D;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR) begin
    if (CLR)
        Q <= 0;
    else
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDC should be placed
-- after architecture statement but before begin keyword

cOMPONENT FDC
    -- synthesis translate_off
    generic (INIT : bit:= '1');
    -- synthesis translate_on
    PORT (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        CLR : in STD_ULOGIC;
        D : in STD_ULOGIC);
end COMPONENT;

-- Component Attribute specification for FDC
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDC_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDC should be placed
-- in architecture after the begin keyword

FDC_INSTANCE_NAME : FDC
    -- synthesis translate_off
generic map (INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
    C => user_C,
    CLR => user_CLR,
    D => user_D);

Verilog Instantiation Template

FDC FDC_instance_name (.Q (user_Q),
                          .C (user_C),
                          .CLR (user_CLR),
                          .D (user_D));

defparam FDC_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDC_1

D Flip-Flop with Negative-Edge Clock and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDC_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDC_1 is a single D-type flip-flop with data input (D), asynchronous clear input (CLR), and data output (Q). The asynchronous CLR, when active, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>CLR</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdc_1 is
begin
  process (C, CLR) begin
    if (CLR = '1') then
      Q <= '0';
    elsif (C' event and C = '0') then
      Q <= D;
    end if;
  end process;
end architecture;
```
end process;
end Behavioral;

Verilog Inference Code
always @ (negedge C or posedge CLR) begin
    if (CLR)
        Q <= 0;
    else
        Q <= D;
end

VHDL Instantiation Template
-- Component Declaration for FDC_1 should be placed
-- after architecture statement but before begin keyword
component FDC_1
    -- synthesis translate_off
    generic (INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
           C : in STD_ULOGIC;
           CLR : in STD_ULOGIC;
           D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDC_1
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of FDC_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDC_1 should be placed
-- in architecture after the begin keyword
FDC_1_INSTANCE_NAME : FDC_1
    -- synthesis translate_off
    generic map (INIT => bit_value)
    -- synthesis translate_on
    port map (Q => user_Q,
              C => user_C,
              CLR => user_CLR,
              D => user_D);

Verilog Instantiation Template
FDC_1 FDC_1_instance_name (.Q (user_Q),
                           .C (user_C),
                           .CLR (user_CLR),
                           .D (user_D));
defparam FDC_1_instance_name.INIT = bit_value;
Commonly Used Constraints

BLKNM, HBLKNM, U_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, and XBLKNM
FDCE

D Flip-Flop with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FDCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDCE is a single D-type flip-flop with clock enable and asynchronous clear. When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data input (D) of FDCE is transferred to the corresponding data output (Q) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data output (Q) Low. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

For XC9500XL and XC9500XV devices, logic connected to the clock enable (CE) input may be implemented using the clock enable product term (p-term) in the macrocell, provided the logic can be completely implemented using the single p-term available for clock enable without requiring feedback from another macrocell. Only FDCE and FDPE flip-flops primitives may take advantage of the clock-enable p-term.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th></th>
<th>CLR</th>
<th>CE</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td></td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>↑</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fdce is
begin
    process (C, CLR) begin
        if (CLR = '1') then
            Q <= '0';
        elsif (C' event and C = '1') then
            if (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @(posedge CLR or posedge C) begin
    if (CLR) 
        Q <= 0;
    else if (CE) 
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDCE should be placed
-- after architecture statement but before begin keyword
component FDCE
    -- synthesis translate_off
    generic (INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    CLR : in STD_ULOGIC;
    D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDCE
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of FDCE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDCE should be placed
-- in architecture after the begin keyword
FDCE_INSTANCE_NAME : FDCE
    -- synthesis translate_off
    generic map (INIT => bit_value)
    -- synthesis translate_on
    port map (Q => user_Q,
    C => user_C,
CE => user_CE,
CLR => user_CLR,
D => user_D);

Verilog Instantiation Template

FDCE FDCE_instance_name (.Q (user_Q),
       .C (user_C),
       .CE (user_CE),
       .CLR (user_CLR),
       .D (user_D));

defparam FDCE_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
**FDCE_1**

D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Clear

**Architectures Supported**

<table>
<thead>
<tr>
<th>FDCE_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-III Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDCE_1 is a single D-type flip-flop with data (D), clock enable (CE), asynchronous clear (CLR) inputs, and data output (Q). The asynchronous CLR input, when High, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop when CLR is Low and CE is High on the High-to-Low clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-III, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Usage**

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

**VHDL Inference Code**

```vhdl
architecture Behavioral of fdce_1 is
begin
  process (C, CLR) begin
    if (CLR = '1') then
      ...  
    end if;
end process;
```

---

FDCE_1

Spartan-II, Spartan-III Primitive
Spartan-3
Virtex, Virtex-E
Virtex-II, Virtex-II Pro, Virtex-II Pro X | Primitive
XC9500, XC9500XV, XC9500XL       | No
CoolRunner XPLA3 | No
CoolRunner-II | No
Q <= '0';
elsif (C' event and C = '0') then
  if (CE = '1') then
    Q <= D;
  end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge CLR or negedge C) begin
  if (CLR)
    Q <= 0;
  else if (CE)
    Q <= D;
  end
end

VHDL Instantiation Template

-- Component Declaration for FDCE_1 should be placed
-- after architecture statement but before begin keyword

component FDCE_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    CLR : in STD_ULOGIC;
    D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDCE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDCE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDCE_1 should be placed
-- in architecture after the begin keyword

FDCE_1_INSTANCE_NAME : FDCE_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
    C => user_C,
    CE => user_CE,
    CLR => user_CLR,
    D => user_D);
Verilog Instantiation Template

FDCE_1 FDCE_1_instance_name (.Q (user_Q),
    .C (user_C),
    .CE (user_CE),
    .CLR (user_CLR),
    .D (user_D));

defparam FDCE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
D Flip-Flop Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDCP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDCP is a single D-type flip-flop with data (D), asynchronous preset (PRE) and clear (CLR) inputs, and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low on the Low-to-High clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdcp is
begin
process (C, CLR, PRE) begin
  if (CLR = '1') then
```

Q <= '0';
elsif (PRE = '1') then
  Q <= '1';
elsif (C' event and C = '1') then
  Q <= D;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge CLR or posedge PRE or posedge C) begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDCP should be placed
-- after architecture statement but before begin keyword

component FDCP
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CLR : in STD_ULOGIC;
       D : in STD_ULOGIC;
       PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDCP
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDCP_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDCP should be placed
-- in architecture after the begin keyword

FDCP_INSTANCE_NAME : FDCP
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            CLR => user_CLR,
            D => user_D,
            PRE => user_PRE);
Verilog Instantiation Template

FDCP FDCP_instance_name (.Q (user_Q),
    .C (user_C),
    .CLR (user_CLR),
    .D (user_D),
    .PRE (user_PRE));

defparam FDCP_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDCP_1

D Flip-Flop with Negative-Edge Clock and Asynchronous Preset and Clear

**Architectures Supported**

<table>
<thead>
<tr>
<th>FDCP_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDCP_1 is a single D-type flip-flop with data (D), asynchronous preset (PRE) and clear (CLR) inputs, and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low on the High-to-Low clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Usage**

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

**VHDL Inference Code**

```vhdl
architecture Behavioral of fdcp_1 is
begin
    process (C, CLR, PRE) begin
        if (CLR = '1') then
            Q <= '0';
        end if;
    end process;
```

elsif (PRE = '1') then
  Q <= '1';
elsif (C' event and C = '0') then
  Q <= D;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge CLR or posedge PRE or negedge C) begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDCP_1 should be placed
-- after architecture statement but before begin keyword

component FDCP_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        CLR : in STD_ULOGIC;
        D : in STD_ULOGIC;
        PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDCP_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDCP_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDCP_1 should be placed
-- in architecture after the begin keyword

FDCP_1_INSTANCE_NAME : FDCP_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            CLR => user_CLR,
            D => user_D,
            PRE => user_PRE);

Verilog Instantiation Template

FDCP_1 FDCP_1_instance_name (.Q (user_Q),
   .C (user_C),
   .CLR (user_CLR),
   .D (user_D),
   .PRE (user_PRE));

defparam FDCP_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDCPE

D Flip-Flop with Clock Enable and Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDCPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDCPE is a single D-type flip-flop with data (D), clock enable (CE), asynchronous preset (PRE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FDCPE Implementation XC9500/XV/XL, CoolRunner XPLA3

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdcpe is
begin
  process (C, CLR, PRE) begin
    if (CLR = '1') then
      Q <= '0';
    elsif (PRE = '1') then
      Q <= '1';
    elsif (C' event and C = '1') then
      if (CE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge CLR or posedge PRE or posedge C) begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (CE)
    Q <= D;
end
```

VHDL Instantiation Template

```
-- Component Declaration for FDCPE should be placed
-- after architecture statement but before begin keyword
```
component FDCPE
  -- synthesis translate_off
  generic (INIT : bit:= '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       CLR : in STD_ULOGIC;
       D : in STD_ULOGIC;
       PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDCPE
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of FDCPE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDCPE should be placed
-- in architecture after the begin keyword
FDCPE_INSTANCE_NAME : FDCPE
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
           C => user_C,
           CE => user_CE,
           CLR => user_CLR,
           D => user_D,
           PRE => user_PRE);

Verilog Instantiation Template
FDCPE FDCPE_instance_name (.Q (user_Q),
                           .C (user_C),
                           .CE (user_CE),
                           .CLR (user_CLR),
                           .D (user_D),
                           .PRE (user_PRE));
defparam FDCPE_instance_name.INIT = bit_value;

Commonly Used Constraints
BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET,
and XBLKNM.
FDCPE_1

D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDCPE_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDCPE_1 is a single D-type flip-flop with data (D), clock enable (CE), asynchronous preset (PRE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the High-to-Low clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fdcpe_1 is
begin
process (C, CLR, PRE) begin
   if (CLR = '1') then
      Q <= '0';
   elsif (PRE = '1') then
      Q <= '1';
   elsif (C' event and C = '0') then
      if (CE = '1') then
         Q <= D;
      end if;
   end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge CLR or posedge PRE or negedge C) begin
   if (CLR)
      Q <= 0;
   else if (PRE)
      Q <= 1;
   else if (CE)
      Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDCPE_1 should be placed
-- after architecture statement but before begin keyword

component FDCPE_1
   -- synthesis translate_off
   generic (INIT : bit := '1');
   -- synthesis translate_on
   port (Q : out STD_ULOGIC;
         C : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         CLR : in STD_ULOGIC;
         D : in STD_ULOGIC;
         PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDCPE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDCPE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDCPE_1 should be placed
-- in architecture after the begin keyword

FDCPE_1_INSTANCE_NAME : FDCPE_1
-- synthesis translate_off
generic map (INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
    C => user_C,
    CE => user_CE,
    CLR => user_CLR,
    D => user_D,
    PRE => user_PRE);

Verilog Instantiation Template

FDCPE_1 FDCPE_1_instance_name (.Q (user_Q),
    .C (user_C),
    .CE (user_CE),
    .CLR (user_CLR),
    .D (user_D),
    .PRE (user_PRE));

defparam FDCPE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
FDD

Dual Edge Triggered D Flip-Flop

Architectures Supported

<table>
<thead>
<tr>
<th>FDD</th>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDD is a single dual edge triggered D-type flip-flop with data input (D) and data output (Q). The data on the D input is loaded into the flip-flop during the Low-to-High and the High-to-Low clock (C) transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

See FDD4,8,16 for information on multiple D flip-flops for CoolRunner-II.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

FDD Implementation CoolRunner-II
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of fdd is
begin
  process (C)
  begin
    if C' event then
      Q <= D;
    end if;
  end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C) begin
  Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDD should be placed
-- after architecture statement but before begin keyword

component FDD
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDD
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDD_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDD should be placed
-- in architecture after the begin keyword

FDD_INSTANCE_NAME : FDD
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            D => user_D);
Verilog Instantiation Template

FDD FDD_instance_name (.Q (user_Q),
 .C (user_C),
 .D (user_D));

defparam FDD_instance_name.INIT = bit_value;
FDD4, FDD8, and FDD16 are multiple dual edge triggered D-type flip-flops with data inputs (D) and data outputs (Q). FDD4, FDD8, and FDD16 are, respectively, 4-bit, 8-bit, and 16-bit registers, each with a common clock (C). The data on the D inputs is loaded into the flip-flop during the Low-to-High and High-to-Low clock (C) transitions.

The flip-flops are asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>FDD4, FDD8, FDD16</th>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDD4, FDD8, FDD16 are multiple dual edge triggered D-type flip-flops with data inputs (D) and data outputs (Q). FDD4, FDD8, and FDD16 are, respectively, 4-bit, 8-bit, and 16-bit registers, each with a common clock (C). The data on the D inputs is loaded into the flip-flop during the Low-to-High and High-to-Low clock (C) transitions.

The flip-flops are asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dz – D0</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

z = 3 for FDD4; z = 7 for FDD8; z = 15 for FDD16
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fdd4 is
begin
    process (C)
    begin
        if C' event then
            Q <= D;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or negedge C) begin
    Q[3:0] <= D[3:0];
end
```
FDD4CE, FDD8CE, FDD16CE

4-, 8-, 16-Bit Dual Edge Triggered Data Registers with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDD4CE, FDD8CE, FDD16CE</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDD4CE, FDD8CE, and FDD16CE are, respectively, 4-, 8-, and 16-bit data registers with clock enable and asynchronous clear. When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data inputs (D) is transferred to the corresponding data outputs (Q) during the Low-to-High and High-to-Low clock (C) transitions. When CLR is High, it overrides all other inputs and resets the data outputs (Q) Low. When CE is Low, clock transitions are ignored.

The flip-flops are asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for FDD4CE; z = 7 for FDD8CE; z = 15 for FDD16CE.
FDD8CE Implementation CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fdd4ce is
begin
    process (C, CLR) begin
        if (CLR = '1') then
            Q <= "0000";
        elsif (C' event) then
            if (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or negedge C or posedge CLR)
begin
    if (CLR)
        Q <= 4'b0;
    else if (CE)
        Q[3:0] <= D[3:0];
end
```
FDD4RE, FDD8RE, FDD16RE

4-, 8-, 16-Bit Dual Edge Triggered Data Registers with Clock Enable and Synchronous Reset

**Architectures Supported**

<table>
<thead>
<tr>
<th>FDD4RE, FDD8RE, FDD16RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDD4RE, FDD8RE, and FDD16RE are, respectively, 4-, 8-, and 16-bit data registers. When the clock enable (CE) input is High, and the synchronous reset (R) input is Low, the data on the data inputs (D) is transferred to the corresponding data outputs (Q0) during the Low-to-High or High-to-Low clock (C) transition. When R is High, it overrides all other inputs and resets the data outputs (Q) Low on the Low-to-High and High-to-Low clock transitions. When CE is Low, clock transitions are ignored.

The flip-flops are asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for FDD4RE; z = 7 for FDD8RE; z = 15 for FDD16RE
FDD8RE Implementation CoolRunner-II

Usage
For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```
architecture Behavioral of fdd4re is
begin
    process (C) begin
        if (C’ event) then
            if (CE = '1') then
                if (R = '1') then
                    Q <= "0000";
                else
                    Q <= D;
                end if;
            end if;
        end if;
    end process;
end Behavioral;
```
Verilog Inference Code

always @ (posedge C or negedge C) begin
    if (R)
        Q <= 4'b0;
    else if (CE)
        Q[3:0] <= D[3:0] ;
end
FDDC

D Dual Edge Triggered Flip-Flop with Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDDC is a single dual edge triggered D-type flip-flop with data (D) and asynchronous clear (CLR) inputs and data output (Q). The asynchronous CLR, when High, overrides all other inputs and sets the Q output Low. The data on the D input is loaded into the flip-flop when CLR is Low on the Low-to-High and High-to-Low clock (C) transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDDC Implementation CoolRunner-II
Usage

This design element typically should be inferred in the design code; however, the
element can be instantiated for cases where strict placement control, relative
placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of fddc is
begin
process (C,CLR) begin
  if (CLR = '1') then
    Q <= '0';
  elsif (C' event) then
    Q <= D;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 1'b0;
  else
    Q <= D ;
end

VHDL Instantiation Template

-- Component Declaration for FDDC should be placed
-- after architecture statement but before begin keyword
compoment FDDC
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        CLR : in STD_ULOGIC;
        D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDC
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of FDDC_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDDC should be placed
-- in architecture after the begin keyword

FDDC_INSTANCE_NAME : FDDC
-- synthesis translate_off
generic map (INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
       C => user_C,
       CLR => user_CLR,
       D => user_D);

Verilog Instantiation Template

FDDC FDDC_instance_name (.Q (user_Q),
                         .C (user_C),
                         .CLR (user_CLR),
                         .D (user_D));

defparam FDDC_instance_name.INIT = bit_value;
FDDCE

Dual Edge Triggered D Flip-Flop with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDDCE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDDCE is a single dual edge triggered D-type flip-flop with clock enable and asynchronous clear. When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data input (D) of FDDCE is transferred to the corresponding data output (Q) during the Low-to-High and High-to-Low clock (C) transitions. When CLR is High, it overrides all other inputs and resets the data output (Q) Low. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

Logic connected to the clock enable (CE) input may be implemented using the clock enable product term (p-term) in the macrocell, provided the logic can be completely implemented using the single p-term available for clock enable without requiring feedback from another macrocell. Only FDDCE and FDDPE flip-flops primitives may take advantage of the clock-enable p-term.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fddce is
begin
process (C, CLR) begin
if (CLR = '1') then
    Q <= '0';
elsif (C' event) then
if (CE = '1') then
    Q <= D;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
begin
    if (CLR)
        Q <= 1'b0;
    else
        if (CE)
            Q <= D;
    end

VHDL Instantiation Template

-- Component Declaration for FDDCE should be placed
-- after architecture statement but before begin keyword

component FDDCE
    port (Q : out STD_ULOGIC;
          C : in STD_ULOGIC;
          CE : in STD_ULOGIC;
          CLR : in STD_ULOGIC;
          D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDCE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for FDDCE should be placed
-- in architecture after the begin keyword

FDDCE_INSTANCE_NAME : FDDCE
port map (Q => user_Q,
            C => user_C,
            CE => user_CE,
            CLR => user_CLR,
            D => user_D);
Verilog Instantiation Template

FDDCE FDDCE_instance_name (.Q (user_Q),
     .C (user_C),
     .CE (user_CE),
     .CLR (user_CLR),
     .D (user_D));
FDDCP

Dual Edge Architectures Supported Triggered D Flip-Flop
Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDDCP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDDCP is a single dual edge triggered D-type flip-flop with data (D), asynchronous preset (PRE) and clear (CLR) inputs, and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low on the Low-to-High and High-to-Low clock (C) transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fddcp is
begin
    process (C, CLR, PRE) begin
        if (CLR = '1') then
```
Q <= '0';
elsif (PRE = '1') then
  Q <= '1';
elsif (C' event) then
  Q <= D;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C or posedge CLR or posedge PRE)
begin
  if (CLR)
    Q <= 1'b0;
  else
    if (PRE)
      Q <= 1'b1;
    else
      Q <= D;
  end
VHDL Instantiation Template

-- Component Declaration for FDDCP should be placed
-- after architecture statement but before begin keyword

component FDDCP
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CLR : in STD_ULOGIC;
       D : in STD_ULOGIC;
       PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDCP
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDDCP_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDDCP should be placed
-- in architecture after the begin keyword

FDDCP_INSTANCE_NAME : FDDCP
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            CLR => user_CLR,
            D => user_D,
PRE => user_PRE);

Verilog Instantiation Template

FDDCP FDDCP_instance_name (.Q (user_Q),
  .C (user_C),
  .CLR (user_CLR),
  .D (user_D),
  .PRE (user_PRE));

defparam FDDCP_instance_name.INIT = bit_value;
FDDCPE

Dual Edge Triggered D Flip-Flop with Clock Enable and Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDDCPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDDCPE is a single dual edge triggered D-type flip-flop with data (D), clock enable (CE), asynchronous preset (PRE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High and High-to-Low clock (C) transitions. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>CLR</th>
<th>PRE</th>
<th>CE</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>No Chg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>↓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>↓</td>
<td>1</td>
</tr>
</tbody>
</table>
FDDCPE Implementation CoolRunner-II

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of fddcpe is
begin
process (C, CLR, PRE) begin
if (CLR = '1') then
    Q <= '0';
elsif (PRE = '1') then
    Q <= '1';
elsif (C' event) then
    if (CE = '1') then
        Q <= D;
    end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR or posedge PRE)
begin
    if (CLR)
        Q <= 1'b0;
    else
        if (PRE)
            Q <= 1'b1;
        else
            if (CE)
                Q <= 1'b1;
        end
    end
end

VHDL Instantiation Template

-- Component Declaration for FDDCPE should be placed
-- after architecture statement but before begin keyword
component FDDCPE
    port (Q : out STD_ULOGIC;
         C : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         CLR : in STD_ULOGIC;
         D : in STD_ULOGIC;
         PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDCPE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter values here

-- Component Instantiation for FDDCPE should be placed
-- in architecture after the begin keyword

FDDCPE_INSTANCE_NAME : FDDCPE
    port map (Q => user_Q,
               C => user_C,
               CE => user_CE,
               CLR => user_CLR,
               D => user_D,
               PRE => user_PRE);

Verilog Instantiation Template

FDDCPE FDDCPE_instance_name (.Q (user_Q),
                                .C (user_C),
                                .CE (user_CE),
                                .CLR (user_CLR),
                                .D (user_D),
                                .PRE (user_PRE));
FDDP

Dual Edge Triggered D Flip-Flop with Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FDDP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td></td>
</tr>
</tbody>
</table>

FDDP is a single dual edge triggered D-type flip-flop with data (D) and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and presets the Q output High. The data on the D input is loaded into the flip-flop when PRE is Low on the Low-to-High and High-to-Low clock (C) transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
</tbody>
</table>

FDDP Implementation CoolRunner-II
Usage

For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fddp is
begin
  process (C,PRE) begin
    if (PRE = '1') then
      Q <= '1';
    elsif (C' event) then
      Q <= D;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or negedge C or posedge PRE)
begin
  if (PRE)
    Q <= 1'b1;
  else
    Q <= D;
end
```

VHDL Instantiation Template

```vhdl
-- Component Declaration for FDDP should be placed after architecture statement but before begin keyword

component FDDP
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
      C : in STD_ULOGIC;
      D : in STD_ULOGIC;
      PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDP
-- should be placed after architecture declaration but before the begin keyword

attribute INIT : string;
attribute INIT of FDDP_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDDP should be placed in architecture after the begin keyword

FDDP_INSTANCE_NAME : FDDP
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
```
port map (Q => user_Q,
    C => user_C,
    D => user_D,
    PRE => user_PRE);

Verilog Instantiation Template

FDDP FDDP_instance_name (.Q (user_Q),
    .C (user_C),
    .D (user_D),
    .PRE (user_PRE));

defparam FDDP_instance_name.INIT = bit_value;
FDDPE

Dual Edge Triggered D Flip-Flop with Clock Enable and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FDDPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDDPE is a single dual edge triggered D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the Q output High. Data on the D input is loaded into the flip-flop when PRE is Low and CE is High on the Low-to-High and High-to-Low clock (C) transitions. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

Logic connected to the clock enable (CE) input may be implemented using the clock enable product term (p-term) in the macrocell, provided the logic can be completely implemented using the single p-term available for clock enable without requiring feedback from another macrocell. Only FDDCE and FDDPE flip-flops primitives may take advantage of the clock-enable p-term.

<table>
<thead>
<tr>
<th>PRE</th>
<th>CE</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>↓</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>↓</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fddpe is
begin
process (C,PRE) begin
if (PRE = '1') then
   Q <= '1';
elif (C' event) then
   if (CE = '1') then
      Q <= D;
   end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge PRE)
begin
if (PRE)
   Q <= 1'b1;
else
   if (CE)
      Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDDPE should be placed
-- after architecture statement but before begin keyword

component FDDPE
   port (Q : out STD_ULOGIC;
         C : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         D : in STD_ULOGIC;
         PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDPE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for FDDPE should be placed
-- in architecture after the begin keyword

FDDPE_INSTANCE_NAME : FDDPE
   port map (Q => user_Q,
             C => user_C,
             CE => user_CE,
             D => user_D,
             PRE => user_PRE);

Verilog Instantiation Template

FDDPE FDDPE_instance_name (.Q (user_Q),
.C (user_C),
.CE (user_CE),
.D (user_D),
.PRE (user_PRE));

defparam FDDPE_instance_name.INIT = bit_value;
FDDR

Dual Edge Triggered D Flip-Flop with Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FDDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDDR is a single dual edge triggered D-type flip-flop with data (D) and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the Low-to-High and High-to-Low clock (C) transitions. The data on the D input is loaded into the flip-flop when R is Low during the Low-to-High or High-to-Low clock transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDDR Implementation CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of fddr is
begin
    process (C) begin
        if (C' event) then
            if (R = '1') then
                Q <= '0';
            else
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C) begin
    if (R) Q <= 1'b0;
    else Q <= D;
end
FDDRCPE

Dual Data Rate D Flip-Flop with Clock Enable and Asynchronous
Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FDDRCPE</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDDRCPE is a Dual Data Rate (DDR) D flip-flop with two separate clocks (C0 and C1) phase shifted 180 degrees that allow selection of two separate data inputs (D0 and D1). It also has clock enable (CE), asynchronous preset (PRE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous PRE, when High, sets the Q output High; CLR, when High, resets the output Low. Data on the D0 input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High C1 clock transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Use the INIT attribute to initialize FDDRCPE during configuration.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>C1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is instantiated rather than inferred.
VHDL Instantiation Template

-- Component Declaration for FDDRCPE should be placed
-- after architecture statement but before begin keyword

component FDDRCPE
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C0 : in STD_ULOGIC;
       C1 : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       CLR : in STD_ULOGIC;
       D0 : in STD_ULOGIC;
       D1 : in STD_ULOGIC;
       PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDRCPE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDDRCPE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDDRCPE should be placed
-- in architecture after the begin keyword

FDDRCPE_INSTANCE_NAME : FDDRCPE
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C0 => user_C0,
            C1 => user_C1,
            CE => user_CE,
            CLR => user_CLR,
            D0 => user_D0,
            D1 => user_D1,
            PRE => user_PRE);
Verilog Instantiation Template

FDDRCPE FDDRCPE_instance_name (.Q (user_Q),
   .C0 (user_C0),
   .C1 (user_C1),
   .CE (user_CE),
   .CLR (user_CLR),
   .D0 (user_D0),
   .D1 (user_D1),
   .PRE (user_PRE));

defparam FDDRCPE_instance_name.INIT = bit_value;
FDDRE

Dual Edge Triggered D Flip-Flop with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FDDRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDDRE is a single dual edge triggered D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the Low-to-High or High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low and CE is High during the Low-to-High and High-to-Low clock transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th></th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### FDDRE Implementation CoolRunner-II

**Usage**

For HDL, this design element can be inferred but not instantiated.

**VHDL Inference Code**

```vhdl
architecture Behavioral of fddre is begin
    process (C) begin
        if (C' event) then
            if (R = '1') then
                Q <= '0';
            elsif (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

**Verilog Inference Code**

```verilog
always @ (posedge C or negedge C) begin
    if (R)
        Q <= 1'b0;
    else
        if (CE)
            Q <= D;
end
```
FDDRRSE

Dual Data Rate D Flip-Flop with Clock Enable and Synchronous Reset and Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDDRRSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDDRRSE is a Dual Data Rate (DDR) D flip-flop with two separate clocks (C0 and C1) phase shifted 180 degrees that allow selection of two separate data inputs (D0 and D1). It also has synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). The reset (R) input, when High, overrides all other inputs and resets the Q output Low during any Low-to-High clock transition (C0 or C1). (Reset has precedence over Set.) When the S input is High and R is Low, the flip-flop is set, output High, during a Low-to-High clock transition (C0 or C1). Data on the D0 input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High C1 clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Use the INIT attribute to initialize FDDRRSE during configuration.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0 C1 CE D0 D1 R S Q</td>
<td></td>
</tr>
<tr>
<td>↑ X X X X X 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>↑ X X X X 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>↑ X X X X 1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>X ↑ X X X 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>X ↑ X X X 1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>X X 0 X X 0 0 No Chg</td>
<td></td>
</tr>
<tr>
<td>↑ X 1 D0 X 0 0 D0</td>
<td></td>
</tr>
<tr>
<td>X ↑ 1 X D1 0 0 D1</td>
<td></td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is instantiated rather than inferred.
VHDL Instantiation Template

-- Component Declaration for FDDRRSE should be placed
-- after architecture statement but before begin keyword

component FDDRRSE
-- synthesis translate_off
generic (INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
    C0 : in STD_ULOGIC;
    C1 : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    D0 : in STD_ULOGIC;
    D1 : in STD_ULOGIC;
    R : in STD_ULOGIC;
    S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDRRSE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDDRRSE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDDRRSE should be placed
-- in architecture after the begin keyword

FDDRRSE_INSTANCE_NAME : FDDRRSE
-- synthesis translate_off
generic map (INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
    C0 => user_C0,
    C1 => user_C1,
    CE => user_CE,
    D0 => user_D0,
    D1 => user_D1,
    R => user_R,
    S => user_S);
Verilog Instantiation Template

FDDRRSE FDDRRSE_instance_name (.Q (user_Q),
  .C0 (user_C0),
  .C1 (user_C1),
  .CE (user_CE),
  .D0 (user_D0),
  .D1 (user_D1),
  .R (user_R),
  .S (user_S));

defparam FDDRRSE_instance_name.INIT = bit_value;
FDDRS

Dual Edge Triggered D Flip-Flop with Synchronous Reset and Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDDRS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDDRS is a single dual edge triggered D-type flip-flop with data (D), synchronous set (S), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High or High-to-Low clock (C) transitions. (Reset has precedence over Set.) When S is High and R is Low, the flip-flop is set, output High, during the Low-to-High or High-to-Low clock transition. When R and S are Low, data on the (D) input is loaded into the flip-flop during the Low-to-High and High-to-Low clock transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDDRS Implementation CoolRunner-II
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fddrs is
begin
    process (C) begin
        if (C' event) then
            if (R = '1') then
                Q <= '0';
            elsif (S = '1') then
                Q <= '1';
            else
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or negedge C)
begin
    if (R)
        Q <= 1'b0;
    else
        if (S)
            Q <= 1'b1;
        else
            Q <= D;
    end
end
```

VHDL Instantiation Template

```vhdl
-- Component Declaration for FDDRS should be placed
-- after architecture statement but before begin keyword

component FDDRS
-- synthesis translate_off
generic (INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    D : in STD_ULOGIC;
    R : in STD_ULOGIC;
    S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDDRS
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
```
attribute INIT of FDDRS_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDDRS should be placed
-- in architecture after the begin keyword

FDDRS_INSTANCE_NAME : FDDRS
-- synthesis translate_off
generic map (INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
  C => user_C,
  D => user_D,
  R => user_R,
  S => user_S);

Verilog Instantiation Template

FDDRS FDDRS_instance_name (.Q (user_Q),
  .C (user_C),
  .D (user_D),
  .R (user_R),
  .S (user_S));

defparam FDDRS_instance_name.INIT = bit_value;
FDDRSE

Dual Edge Triggered D Flip-Flop with Synchronous Reset and Set and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FDDRSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDDRSE is a single dual edge triggered D-type flip-flop with synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). The reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High or High-to-Low clock transitions. (Reset has precedence over Set.) When the set (S) input is High and R is Low, the flip-flop is set, output High, during the Low-to-High or High-to-Low clock (C) transition. Data on the D input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High and High-to-Low clock transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FDDRSE Implementation CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fddrse is
begin
  process (C) begin
    if (C' event) then
      if (R = '1') then
        Q <= '0';
      elsif (S = '1') then
        Q <= '1';
      elsif (CE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or negedge C)
begin
  if (R)
    Q <= 1'b0;
  else
    if (S)
      Q <= 1'b1;
    else
      if (CE)
        Q <= D;
  end
```

---

**Diagram**

[Diagram of FDDRSE Implementation in CoolRunner-II]
FDDS

Dual Edge Triggered D Flip-Flop with Synchronous Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDDS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDDS is a single dual edge triggered D-type flip-flop with data (D) and synchronous set (S) inputs and data output (Q). The synchronous set input, when High, sets the Q output High on the Low-to-High or High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low during the Low-to-High and High-to-Low clock (C) transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDDS Implementation CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of fdds is
begin
process (C) begin
  if (C' event) then
    if (S = '1') then
      Q <= '1';
    else
      Q <= D;
    end if;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
  if (S)
    Q <= 1'b1;
  else
    Q <= D;
end
FDDSE

D Flip-Flop with Clock Enable and Synchronous Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDDSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDDSE is a single dual edge triggered D-type flip-flop with data (D), clock enable (CE), and synchronous set (S) inputs and data output (Q). The synchronous set (S) input, when High, overrides the clock enable (CE) input and sets the Q output High during the Low-to-High or High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low and CE is High during the Low-to-High and High-to-Low clock (C) transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
FDDSE Implementation CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of fddse is
begin
  process (C) begin
    if (C' event) then
      if (S = '1') then
        Q <= '1';
      elseif (CE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C) begin
  if (S)
    Q <= 1'b1;
  else
    if (CE)
      Q <= D;
  end
FDDSR

Dual Edge Triggered D Flip-Flop with Synchronous Set and Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FDDSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDDSR is a single dual edge triggered D-type flip-flop with data (D), synchronous reset (R) and synchronous set (S) inputs and data output (Q). When the set (S) input is High, it overrides all other inputs and sets the Q output High during the Low-to-High or High-to-Low clock transition. (Set has precedence over Reset.) When reset (R) is High and S is Low, the flip-flop is reset, output Low, on the Low-to-High or High-to-Low clock transition. Data on the D input is loaded into the flip-flop when S and R are Low on the Low-to-High and High-to-Low clock transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDDSR Implementation CoolRunner-II
Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of fddsr is
begin
  process (C) begin
    if (C' event) then
      if (S = '1') then
        Q <= '1';
      elseif (R = '1') then
        Q <= '0';
      else
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
  if (S)
    Q <= 1'b1;
  else
    if (R)
      Q <= 1'b0;
    else
      Q <= D;
  end
end
FDDSRE

Dual Edge Triggered D Flip-Flop with Synchronous Set and Reset and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>FDDSRE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDDSRE is a single dual edge triggered D-type flip-flop with synchronous set (S), synchronous reset (R), and clock enable (CE) inputs and data output (Q). When synchronous set (S) is High, it overrides all other inputs and sets the Q output High during the Low-to-High or High-to-Low clock transition. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, output Q is reset Low during the Low-to-High or High-to-Low clock transition. Data is loaded into the flip-flop when S and R are Low and CE is High during the Low-to-High and High-to-Low clock transitions. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated in Verilog by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S</strong></td>
<td><strong>R</strong></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FDDSRE Implementation CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fddsre is
begin
    process (C) begin
        if (C' event) then
            if (S = '1') then
                Q <= '1';
            elsif (R = '1') then
                Q <= '0';
            elsif (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or negedge C)
begin
    if (S)
        Q <= 1'b1;
    else
        if (R)
            Q <= 1'b0;
        else
            if (CE)
                Q <= D;
end
```
FDE

D Flip-Flop with Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>FDE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDE is a single D-type flip-flop with data input (D), clock enable (CE), and data output (Q). When clock enable is High, the data on the D input is loaded into the flip-flop during the Low-to-High clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fde is
begin
    process (C)
    begin
        if (C' event and C='1') then
            if (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end architecture;
```
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C) begin
  if (CE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDE should be placed
-- after architecture statement but before begin keyword

component FDE
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDE should be placed
-- in architecture after the begin keyword

FDE_INSTANCE_NAME : FDE
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            CE => user_CE,
            D => user_D);

Verilog Instantiation Template

FDE FDE_instance_name (.Q (user_Q),
                       .C (user_C),
                       .CE (user_CE),
                       .D (user_D));

defparam FDE_instance_name.INIT = bit_value;
Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
FDE_1

D Flip-Flop with Negative-Edge Clock and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>FDE_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDE_1 is a single D-type flip-flop with data input (D), clock enable (CE), and data output (Q). When clock enable is High, the data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fde_1 is begin
    process (C)
    begin
        if (C' event and C='0') then
            if (CE = '1') then Q <= D;
        end if;
    end if;
end if;
```
end process;
end Behavioral;

Verilog Inference Code

always @(negedge C) begin
  if (CE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDE_1 should be placed
-- after architecture statement but before begin keyword

component FDE_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       D : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of
  FDE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDE_1 should be placed
-- in architecture after the begin keyword

FDE_1INSTANCE_NAME : FDE_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
           C => user_C,
           CE => user_CE,
           D => user_D);

Verilog Instantiation Template

FDE_1 FDE_1_instance_name (.Q (user_Q),
                           .C (user_C),
                           .CE (user_CE),
                           .D (user_D));

defparam FDE_1_instance_name.INIT = bit_value;
Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
FDP

D Flip-Flop with Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDP is a single D-type flip-flop with data (D) and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and presets the Q output High. The data on the D input is loaded into the flip-flop when PRE is Low on the Low-to-High clock (C) transition.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The active level of the GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
</tbody>
</table>
FDP Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdp is
begin
process (C,PRE) begin
  if (PRE = '1') then
    Q <= '1';
  elsif (C' event and C = '1') then
    Q <= D;
  end if;
end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge PRE or posedge C) begin
  if (PRE)
    Q <= 1;
  else
    Q <= D;
end
```

VHDL Instantiation Template

```vhdl
-- Component Declaration for FDP should be placed
-- after architecture statement but before begin keyword
component FDP
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       D : in STD_ULOGIC;
       PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDP
```
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDP_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDP should be placed
-- in architecture after the begin keyword

FDP_INSTANCE_NAME : FDP
    -- synthesis translate_off
    generic map (    
        INIT => bit_value)
    -- synthesis translate_on
    port map (Q => user_Q, 
        C => user_C, 
        D => user_D, 
        PRE => user_PRE);

Verilog Instantiation Template

FDP FDP_instance_name (.Q (user_Q), 
     .C (user_C), 
     .D (user_D), 
     .PRE (user_PRE));

defparam FDP_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDP_1

D Flip-Flop with Negative-Edge Clock and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FDP_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>No</td>
</tr>
</tbody>
</table>

FDP_1 is a single D-type flip-flop with data (D) and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and presets the Q output High. The data on the D input is loaded into the flip-flop when PRE is Low on the High-to-Low clock (C) transition.

The flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The active level of the GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```
architecture Behavioral of fdp_1 is
begin
    process (C, PRE) begin
        if (PRE = '1') then
            Q <= '1';
        elsif (C' event and C = '0') then
            Q <= D;
        end if;
    end process;
```

Libraries Guide
ISE 6.li
www.xilinx.com
1-800-255-7778
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge PRE or negedge C) begin
    if (PRE)
        Q <= 1;
    else
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDP_1 should be placed
-- after architecture statement but before begin keyword

cOMPONENT FDP_1
    -- synthesis translate_off
    GENERIC (
        INIT : BIT := '1');
    -- synthesis translate_on
    PORT (Q : OUT STD_ULOGIC;
         C : IN STD_ULOGIC;
         D : IN STD_ULOGIC;
         PRE : IN STD_ULOGIC);
END COMPONENT;

-- Component Attribute specification for FDP_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : STRING;
attribute INIT of FDP_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDP_1 should be placed
-- in architecture after the begin keyword

FDP_1_INSTANCE_NAME : FDP_1
    -- synthesis translate_off
    GENERIC MAP (
        INIT => bit_value)
    -- synthesis translate_on
    PORT MAP (Q => user_Q,
        C => user_C,
        D => user_D,
        PRE => user_PRE);

Verilog Instantiation Template

FDP_1 FDP_1_instance_name (.Q (user_Q),
    .C (user_C),
    .D (user_D),
    .PRE (user_PRE));

defparam FDP_1_instance_name.INIT = bit_value;
Commonly Used Constraints
BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDPE

D Flip-Flop with Clock Enable and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FDPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FDPE is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the Q output High. Data on the D input is loaded into the flip-flop when PRE is Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

For XC9500XL and XC9500XV devices, logic connected to the clock enable (CE) input may be implemented using the clock enable product term (p-term) in the macrocell, provided the logic can be completely implemented using the single p-term available for clock enable without requiring feedback from another macrocell. Only FDCE and FDPE flip-flop primitives may take advantage of the clock-enable p-term.

For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied. These devices simulate power-on when global set/reset (GSR) is active.

The active level of the GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of fdpe is
begin
    process (C,PRE) begin
        if (PRE = '1') then
            Q <= '1';
        elsif (C' event and C = '1') then
            if (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (posedge PRE or posedge C) begin
    if (PRE)
        Q <= 1;
    else if (CE)
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDPE should be placed
-- after architecture statement but before begin keyword

component FDPE
    -- synthesis translate_off
    generic (
        INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
          C : in STD_ULOGIC;
          CE : in STD_ULOGIC;
          D : in STD_ULOGIC;
          PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDPE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of

FDPE_instance_name : label is "0";
-- values can be (0 or 1)
Component Instantiation for FDPE should be placed
in architecture after the begin keyword.
```
FDPE_INSTANCE_NAME : FDPE
  -- synthesis translate_off
  generic map (       
    INIT => bit_value) 
  -- synthesis translate_on 
  port map (Q => user_Q,       
             C => user_C,       
             CE => user_CE,       
             D => user_D,       
             PRE => user_PRE); 
```

Verilog Instantiation Template
```
FDPE FDPE_instance_name (.Q (user_Q),      
  .C (user_C),      
  .CE (user_CE),      
  .D (user_D),      
  .PRE (user_PRE)); 
```
defparam
```
FDPE_instance_name.INIT = bit_value;
```

Commonly Used Constraints
BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, and XBLKNM
FDPE_1

D Flip-Flop with Negative-Edge Clock, Clock Enable, and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FDPE_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDPE_1 is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the Q output High. Data on the D input is loaded into the flip-flop when PRE is Low and CE is High on the High-to-Low clock (C) transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-III, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The active level of the GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdpe_1 is
begin
  process (C,PRE) begin
    if (PRE = '1') then
```
Q <= '1';
elsif (C' event and C = '0') then
  if (CE = '1') then
    Q <= D;
  end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge PRE or negedge C) begin
  if (PRE)
    Q <= 1;
  else if (CE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDPE_1 should be placed
-- after architecture statement but before begin keyword

component FDPE_1
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    D : in STD_ULOGIC;
    PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDPE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDPE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDPE_1 should be placed
-- in architecture after the begin keyword --

FDPE_1_INSTANCE_NAME : FDPE_1
  -- synthesis translate_off
  generic map (
    INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
    C => user_C,
    CE => user_CE,
    D => user_D,
    PRE => user_PRE);
Verilog Instantiation Template

FDPE_1 $FDPE_1$ Instance_Name (.Q (user_Q),
   .C (user_C),
   .CE (user_CE),
   .D (user_D),
   .PRE (user_PRE));

defparam FDPE_1_Instance_Name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
D Flip-Flop with Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FDR</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDR is a single D-type flip-flop with data (D) and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R D C</td>
<td>Q</td>
</tr>
<tr>
<td>1 X</td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td></td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdr is begin
process (C) begin
  if (C' event and C = '1') then
```
if (R = '1') then  
  Q <= '0';  
else  
  Q <= D; 
end if;  
end if;  
end process;  
end Behavioral;

Verilog Inference Code

always @(posedge C) begin  
  if (R)  
    Q <= 0;  
  else  
    Q <= D; 
end

VHDL Instantiation Template

-- Component Declaration for FDR should be placed  
-- after architecture statement but before begin keyword

component FDR  
  -- synthesis translate_off 
  generic (  
    INIT : bit := '1');  
  -- synthesis translate_on 
  port (Q : out STD_ULOGIC;  
        C : in STD_ULOGIC;  
        D : in STD_ULOGIC;  
        R : in STD_ULOGIC); 
end component; 

-- Component Attribute specification for FDR  
-- should be placed after architecture declaration but  
-- before the begin keyword

attribute INIT : string;  
attribute INIT of FDR_instance_name : label is "0";  
-- values can be (0 or 1)

-- Component Instantiation for FDR should be placed  
-- in architecture after the begin keyword

FDR_INSTANCE_NAME : FDR  
  -- synthesis translate_off 
  generic map (  
    INIT => bit_value)  
  -- synthesis translate_on 
  port map (Q => user_Q,  
            C => user_C,  
            D => user_D,  
            R => user_R);
Verilog Instantiation Template

FDR FDR_instance_name (.Q (user_Q),
  .C (user_C),
  .D (user_D),
  .R (user_R));

defparam FDR_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDR_1

D Flip-Flop with Negative-Edge Clock and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FDR_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDR_1 is a single D-type flip-flop with data (D) and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low during the High-to-Low clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdr_1 is
begin
process (C) begin
if (C' event and C = '0') then
if (R = '1') then
  Q <= '0';
else
  Q <= '0';
end if;
end if;
end process;
end Behavioral;
```
Q <= D;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (negedge C) begin
  if (R)
    Q <= 0;
  else
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDR_1 should be placed
-- after architecture statement but before begin keyword

component FDR_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        D : in STD_ULOGIC;
        R : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDR_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDR_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDR_1 should be placed
-- in architecture after the begin keyword

FDR_1_INSTANCE_NAME : FDR_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            D => user_D,
            R => user_R);
Verilog Instantiation Template

FDR_1 FDR_1_instance_name (.Q (user_Q),
   .C (user_C),
   .D (user_D),
   .R (user_R));

defparam FDR_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDRE

D Flip-Flop with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FDRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDRE is a single D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low and CE is High during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
FDRE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```
architecture Behavioral of fdre is
begin
  process (C) begin
    if (C' event and C = '1') then
      if (R = '1') then
        Q <= '0';
      elsif (CE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```
always @(posedge C) begin
  if (R)
    Q <= 0;
  else if (CE)
    Q <= D;
end
```

VHDL Instantiation Template

```
-- Component Declaration for FDRE should be placed
-- after architecture statement but before begin keyword

component FDRE
  -- synthesis translate_off
  generic (  
```
INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    D : in STD_ULOGIC;
    R : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDRE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDRE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDRE should be placed
-- in architecture after the begin keyword

FDRE_INSTANCE_NAME : FDRE
    -- synthesis translate_off
generic map (    INIT => bit_value)
    -- synthesis translate_on
port map (Q => user_Q,
    C => user_C,
    CE => user_CE,
    D => user_D,
    R => user_R);

Verilog Instantiation Template

FDRE FDRE_instance_name (.Q (user_Q),
    .C (user_C),
    .CE (user_CE),
    .D (user_D),
    .R (user_R));

defparam FDRE_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
FDRE_1

D Flip-Flop with Negative-Clock Edge, Clock Enable, and Synchronous Reset

**Architectures Supported**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDRE_1 is a single D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low on the High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when R is Low and CE is High during the High-to-Low clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Usage**

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

**VHDL Inference Code**

```vhdl
architecture Behavioral of fdre_1 is
begin
    process (C) begin
        if (C' event and C = '0') then
            -- Code
        end if;
    end process;
end Behavioral;
```
if (R = '1') then
  Q <= '0';
elsif (CE = '1') then
  Q <= D;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (negedge C) begin
  if (R)
    Q <= 0;
  else if (CE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDRE_1 should be placed
-- after architecture statement but before begin keyword

component FDRE_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        CE : in STD_ULOGIC;
        D : in STD_ULOGIC;
        R : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDRE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDRE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDRE_1 should be placed
-- in architecture after the begin keyword

FDRE_1_INSTANCE_NAME : FDRE_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            C => user_C,
            CE => user_CE,
            D => user_D,
            R => user_R);
Verilog Instantiation Template

FDRE_1 FDRE_1_instance_name (.Q (user_Q),
  .C (user_C),
  .CE (user_CE),
  .D (user_D),
  .R (user_R));

defparam FDRE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
**FDRS**

**D Flip-Flop with Synchronous Reset and Set**

**Architectures Supported**

<table>
<thead>
<tr>
<th>FDRS</th>
<th>Primitive</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td></td>
</tr>
</tbody>
</table>

FDRS is a single D-type flip-flop with data (D), synchronous set (S), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock (C) transition. (Reset has precedence over Set.) When S is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock transition. When R and S are Low, data on the (D) input is loaded into the flip-flop during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FDRS Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdrs is
begin
  process (C)
  begin
    if (C' event and C = '1') then
      if (R = '1') then
        Q <= '0';
      elsif (S = '1') then
        Q <= '1';
      else
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C) begin
  if (R)
    Q <= 0;
  else if (S)
    Q <= 1;
  else
    Q <= D;
end
```

VHDL Instantiation Template

```vhdl
-- Component Declaration for FDRS should be placed
-- after architecture statement but before begin keyword

component FDRS
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC);
end component;
```

---

D
R
S
C
AND2B1
OR2
FD
Q
---
C : in STD_ULOGIC;
D : in STD_ULOGIC;
R : in STD_ULOGIC;
S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDRS
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDRS_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDRS should be placed
-- in architecture after the begin keyword

FDRS_INSTANCE_NAME : FDRS
-- synthesis translate_off
genric map (
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
  C => user_C,
  D => user_D,
  R => user_R,
  S => user_S);

Verilog Instantiation Template

FDRS FDRS_instance_name (.Q (user_Q),
  .C (user_C),
  .D (user_D),
  .R (user_R),
  .S (user_S));

defparam FDRS_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
**FDRS_1**

**D Flip-Flop with Negative-Clock Edge and Synchronous Reset and Set**

**Architectures Supported**

<table>
<thead>
<tr>
<th>FDRS_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDRS_1 is a single D-type flip-flop with data (D), synchronous set (S), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the Q output Low during the High-to-Low clock (C) transition. (Reset has precedence over Set.) When S is High and R is Low, the flip-flop is set, output High, during the High-to-Low clock transition. When R and S are Low, data on the (D) input is loaded into the flip-flop during the High-to-Low clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Usage**

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fdrs_1 is
begin
    process (C) begin
        if (C' event and C = '0') then
            if (R = '1') then
                Q <= '0';
            elsif (S = '1') then
                Q <= '1';
            else
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (negedge C) begin
    if (R)
        Q <= 0;
    else if (S)
        Q <= 1;
    else
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDRS_1 should be placed after architecture statement but before begin keyword

component FDRS_1
    -- synthesis translate_off
    generic (
        INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        D : in STD_ULOGIC;
        R : in STD_ULOGIC;
        S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDRS_1 should be placed after architecture declaration but before the begin keyword

attribute INIT : string;
attribute INIT of FDRS_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDRS_1 should be placed in architecture after the begin keyword

FDRS_1_INSTANCE_NAME : FDRS_1
-- synthesis translate_off
generic map (  
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
   C => user_C,
   D => user_D,
   R => user_R,
   S => user_S);

Verilog Instantiation Template

FDRS_1 FDRS_1_instance_name (.Q (user_Q),  
   .C (user_C),  
   .D (user_D),  
   .R (user_R),  
   .S (user_S));

defparam FDRS_1_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
FDRSE is a single D-type flip-flop with synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). The reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock transition. (Reset has precedence over Set.) When the set (S) input is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock (C) transition. Data on the D input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>CE</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>C</td>
<td>Q</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>No Chg</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
</tr>
</tbody>
</table>
FDRSE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fdrse is
begin
    process (C) begin
        if (C' event and C = '1') then
            if (R = '1') then
                Q <= '0';
            elsif (S = '1') then
                Q <= '1';
            elsif (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C) begin
    if (R)
        Q <= 0;
    else if (S)
        Q <= 1;
    else if (CE)
        Q <= D;
end
```
VHDL Instantiation Template

-- Component Declaration for FDRSE should be placed
-- after architecture statement but before begin keyword

component FDRSE
  -- synthesis translate_off
generic (
    INIT : bit := '1');
  -- synthesis translate_on
port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    D : in STD_ULOGIC;
    R : in STD_ULOGIC;
    S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDRSE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDRSE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDRSE should be placed
-- in architecture after the begin keyword

FDRSE_INSTANCE_NAME : FDRSE
  -- synthesis translate_off
generic map (
    INIT => bit_value)
  -- synthesis translate_on
port map (Q => user_Q,
    C => user_C,
    CE => user_CE,
    D => user_D,
    R => user_R,
    S => user_S);

Verilog Instantiation Template

FDRSE FDRSE_instance_name (.Q (user_Q),
    .C (user_C),
    .CE (user_CE),
    .D (user_D),
    .R (user_R),
    .S (user_S));

defparam FDRSE_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET and XBLKNM
FDRSE_1

D Flip-Flop with Negative-Clock Edge, Synchronous Reset and Set, and Clock Enable

FDRSE_1 is a single D-type flip-flop with synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). The reset (R) input, when High, overrides all other inputs and resets the Q output Low during the High-to-Low clock transition. (Reset has precedence over Set.) When the set (S) input is High and R is Low, the flip-flop is set, output High, during the High-to-Low clock (C) transition. Data on the D input is loaded into the flip-flop when R and S are Low and CE is High during the High-to-Low clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R S CE D C</td>
<td>Q</td>
</tr>
<tr>
<td>1 X X</td>
<td>↓ 0</td>
</tr>
<tr>
<td>0 1 X</td>
<td>↓ 1</td>
</tr>
<tr>
<td>0 0 X</td>
<td>↓ No Chg</td>
</tr>
<tr>
<td>0 0 1</td>
<td>↓ 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>↓ 0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fdrse_1 is
begin
    process (C) begin
        if (C' event and C = '0') then
            if (R = '1') then
                Q <= '0';
            elsif (S = '1') then
                Q <= '1';
            elsif (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (negedge C) begin
    if (R)
        Q <= 0;
    else if (S)
        Q <= 1;
    else
        Q => D;
end

VHDL Instantiation Template

-- Component Declaration for FDRSE_1 should be placed
-- after architecture statement but before begin keyword

component FDRSE_1
    -- synthesis translate_off
generic (
        INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        CE : in STD_ULOGIC;
        D : in STD_ULOGIC;
        R : in STD_ULOGIC;
        S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDRSE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDRSE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDRSE_1 should be placed
-- in architecture after the begin keyword
FDRSE_1

FDRSE_1_INSTANCE_NAME : FDRSE_1
-- synthesis translate_off
generic map (    
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,    
  C => user_C,    
  CE => user_CE,    
  D => user_D,    
  R => user_R,    
  S => user_S);

Verilog Instantiation Template

FDRSE_1 FDRSE_1_instance_name (.Q (user_Q),    
  .C (user_C),    
  .CE (user_CE),    
  .D (user_D),    
  .R (user_R),    
  .S (user_S));
defparam FDRSE_1_instance_name.INIT = bit_value;

Commonly Used Constraints
  BLKNM
  HBLKNM
  HU_SET
  INIT
  IOB
  LOC
  REG
  RLOC
  TIMEGRP
  TNM
  U_SET
  XBLKNM
FDS

D Flip-Flop with Synchronous Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FDS is a single D-type flip-flop with data (D) and synchronous set (S) inputs and data output (Q). The synchronous set input, when High, sets the Q output High on the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low during the Low-to-High clock (C) transition.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied. For all other devices (XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II), the flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FDS will set when GSR is active. For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fds is
begin
process (C) begin
  if (C' event and C = '1') then
    if (S = '1') then
      Q <= '1';
    else
      Q <= D;
    end if;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C) begin
  if (S)
    Q <= 1;
  else
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDS should be placed
-- after architecture statement but before begin keyword

cOMPONENT FDS
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    C : in STD_ULOGIC;
    D : in STD_ULOGIC;
    S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDS
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDS_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDS should be placed
-- in architecture after the begin keyword

FDS_INSTANCE_NAME : FDS
  -- synthesis translate_off
  generic map (
    INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
C => user_C,
D => user_D,
S => user_S);

Verilog Instantiation Template

FDS FDS_instance_name (.Q (user_Q),
    .C (user_C),
    .D (user_D),
    .S (user_S));

defparam FDS_instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDS_1

D Flip-Flop with Negative-Edge Clock and Synchronous Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDS_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDS_1 is a single D-type flip-flop with data (D) and synchronous set (S) inputs and data output (Q). The synchronous set input, when High, sets the Q output High on the High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low during the High-to-Low clock (C) transition.

For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FDS_1 will set when GSR is active. For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>S</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>‾</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>‾</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>‾</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of fds_1 isegin{verbatim}
begin
    process (C) begin
        if (C’ event and C = '0') then
```
if (S = '1') then
  Q <= '1';
else
  Q <= D;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (negedge C) begin
  if (S)
    Q <= 1;
  else
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDS_1 should be placed
-- after architecture statement but before begin keyword

component FDS_1

   -- synthesis translate_off
   generic (  
     INIT : bit := '1');
   -- synthesis translate_on
   port (Q : out STD_ULOGIC;
      C : in STD_ULOGIC;
      D : in STD_ULOGIC;
      S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDS_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDS_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDS_1 should be placed
-- in architecture after the begin keyword

FDS_1_INSTANCE_NAME : FDS_1

   -- synthesis translate_off
   generic map (  
     INIT => bit_value)
   -- synthesis translate_on
   port map (Q => user_Q,
      C => user_C,
      D => user_D,
      S => user_S);
Verilog Instantiation Template

```verilog
FDS_1 FDS_1_instance_name (.Q (user_Q),
    .C (user_C),
    .D (user_D),
    .S (user_S));

defparam FDS_1_instance_name.INIT = bit_value;
```

Commonly Used Constraints

- BLKNM
- HBLKNM
- HU_SET
- INIT
- IOB
- LOC
- REG
- RLOC
- TIMEGRP
- TNM
- U_SET
- XBLKNM
FDSE

D Flip-Flop with Clock Enable and Synchronous Set

Architectures Supported

<table>
<thead>
<tr>
<th>FDSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDSE is a single D-type flip-flop with data (D), clock enable (CE), and synchronous set (S) inputs and data output (Q). The synchronous set (S) input, when High, overrides the clock enable (CE) input and sets the Q output High during the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low and CE is High during the Low-to-High clock (C) transition.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied.

For all other devices (XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II), the flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FDSE will set when GSR is active. For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
**FDSE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II**

**Usage**

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

**VHDL Inference Code**

```vhdl
architecture Behavioral of fdse is
begin
    process (C) begin
        if (C' event and C = '1') then
            if (S = '1') then
                Q <= '1';
            elsif (CE = '1') then
                Q <= D;
            end if;
        end if;
    end process;
end Behavioral;
```

**Verilog Inference Code**

```verilog
always @ (posedge C) begin
    if (S)
        Q <= 1;
    else if (CE)
        Q <= D;
end
```
VHDL Instantiation Template

-- Component Declaration for FDSE should be placed
-- after architecture statement but before begin keyword

component FDSE
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       D : in STD_ULOGIC;
       S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDSE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDSE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDSE should be placed
-- in architecture after the begin keyword

FDSE_INSTANCE_NAME : FDSE
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
           C => user_C,
           CE => user_CE,
           D => user_D,
           S => user_S);

Verilog Instantiation Template

FDSE FDSE_instance_name (.Q (user_Q),
                        .C (user_C),
                        .CE (user_CE),
                        .D (user_D),
                        .S (user_S));

defparam FDSE_instance_name.INIT = bit_value;
Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
I0B
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDSE_1

D Flip-Flop with Negative-Edge Clock, Clock Enable, and Synchronous Set

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FDSE_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

FDSE_1 is a single D-type flip-flop with data (D), clock enable (CE), and synchronous set (S) inputs and data output (Q). The synchronous set (S) input, when High, overrides the clock enable (CE) input and sets the Q output High during the High-to-Low clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low and CE is High during the High-to-Low clock (C) transition.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FDSE_1 will set when GSR is active. For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of fdse_1 is
begin
  process (C) begin
    if (C' event and C = '0') then
      if (S = '1') then
        Q <= '1';
      elsif (CE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;

Verilog Inference Code

always @(negedge C) begin
  if (S)
    Q <= 1;
  else if (CE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for FDSE_1 should be placed
-- after architecture statement but before begin keyword

component FDSE_1
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        C : in STD_ULOGIC;
        CE : in STD_ULOGIC;
        D : in STD_ULOGIC;
        S : in STD_ULOGIC);
end component;

-- Component Attribute specification for FDSE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of FDSE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for FDSE_1 should be placed
-- in architecture after the begin keyword

FDSE_1_INSTANCE_NAME : FDSE_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
port map (Q => user_Q,
    C => user_C,
    CE => user_CE,
    D => user_D,
    S => user_S);

Verilog Instantiation Template

FDSE_1 instance_name (.Q (user_Q),
    .C (user_C),
    .CE (user_CE),
    .D (user_D),
    .S (user_S));

defparam instance_name.INIT = bit_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FDSR

D Flip-Flop with Synchronous Set and Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FDSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDSR is a single D-type flip-flop with data (D), synchronous reset (R) and synchronous set (S) inputs and data output (Q). When the set (S) input is High, it overrides all other inputs and sets the Q output High during the Low-to-High clock transition. (Set has precedence over Reset.) When reset (R) is High and S is Low, the flip-flop is reset, output Low, on the Low-to-High clock transition. Data on the D input is loaded into the flip-flop when S and R are Low on the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FDSR Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of fdsr is
begin
  process (C) begin
    if (C' event and C = '1') then
      if (S = '1') then
        Q <= '1';
      elsif (R = '1') then
        Q <= '0';
      else
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;

Verilog Inference Code

always @(posedge C) begin
  if (S)
    Q <= 1;
  else if (R)
    Q <= 0;
  else
    Q <= D;
end
FDSRE

D Flip-Flop with Synchronous Set and Reset and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>FDSRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FDSRE is a single D-type flip-flop with synchronous set (S), synchronous reset (R), and clock enable (CE) inputs and data output (Q). When synchronous set (S) is High, it overrides all other inputs and sets the Q output High during the Low-to-High clock transition. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, output Q is reset Low during the Low-to-High clock transition. Data is loaded into the flip-flop when S and R are Low and CE is High during the Low-to-high clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FDSRE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of fdsre is
begin
  process (C) begin
    if (C' event and C = '1') then
      if (S = '1') then
        Q <= '1';
      elsif (R = '1') then
        Q <= '0';
      elsif (CE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C) begin
  if (S)
    Q <= 1;
  else if (R)
    Q <= 0;
  else if (CE)
    Q <= D;
end
FJKC

J-K Flip-Flop with Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FJKC</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FJKC is a single J-K-type flip-flop with J, K, and asynchronous clear (CLR) inputs and data output (Q). The asynchronous clear (CLR) input, when High, overrides all other inputs and resets the Q output Low. When CLR is Low, the output responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>J</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of fjkc is
begin
process (C, CLR)
begin
if (CLR='1') then
Q <= '0';
elseif (C'event and C='1') then
if (J='0') then
if (K='1') then
Q <= '0';
end if;
else
if (K='0') then
Q <= '1';
else
Q <= not Q;
end if;
end if;
each (K='0') then
Q <= '1';
each (K='1') then
Q <= '0';
deif;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edifi;
edi
end if;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or posedge CLR)
beg
if (CLR)
  Q <= 0;
else
  begin
    if (!J)
      begin
        if (K)
          Q <= 0;
        else
          begin
            if (!K)
              Q <= 1;
            else
              Q <= !Q;
          end
      end
  end
end
J-K Flip-Flop with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FJKCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FJKCE is a single J-K-type flip-flop with J, K, clock enable (CE), and asynchronous clear (CLR) inputs and data output (Q). The asynchronous clear (CLR), when High, overrides all other inputs and resets the Q output Low. When CLR is Low and CE is High, Q responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock transition. When CE is Low, the clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fjkce is
begin
process (C, CLR)
begin
if (CLR='1') then
  Q <= '0';
elsif (C'event and C='1') then
  Q <= OR3;
end process;
end architecture;
```

FJKCE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

if (CE='1') then
    if (J='0') then
        if (K='1') then
            Q <= '0';
        end if;
    else
        if (K='0') then
            Q <= '1';
        else
            Q <= not Q;
        end if;
    end if;
end if;
end process;
end Behavioral;

Verilog Inference Code
always @(posedge C or posedge CLR)
begin
    if (CLR)
        Q <= 0;
    else if (CE)
        begin
            if (!J)
                begin
                    if (K)
                        Q <= 0;
                    else
                        begin
                            if (!K)
                                Q <= 1;
                            else
                                Q <= !Q;
                        end
                end
        end
end
FJKCP

J-K Flip-Flop with Asynchronous Clear and Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FJKCP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FJKCP is a single J-K-type flip-flop with J, K, asynchronous clear (CLR), and asynchronous preset (PRE) inputs and data output (Q). When the asynchronous preset (PRE) is High and CLR is Low, all other inputs are ignored and Q is set High. The asynchronous preset (PRE), when High, and CLR set to Low overrides all other inputs and sets the Q output High. When CLR and PRE are Low, Q responds to the state of the J and K inputs during the Low-to-High clock transition, as shown in the following truth table.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FJKCP Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fjkcp is
begin
process (C, CLR, PRE)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (PRE='1') then
    Q <= '1';
  elsif (C'event and C='1') then
    if (J='0') then
      if (K='1') then
        Q <= '0';
      end if;
    else
      if (K='0') then
        Q <= '1';
      else
        Q <= not Q;
      end if;
    end if;
  end if;
end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or posedge CLR or posedge PRE)
begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
```
else
begin
  if (!J)
  begin
    if (K) Q <= 0;
    end
  else
  begin
    if (!K) Q <= 1;
    else Q <= !Q;
  end
end
J-K Flip-Flop with Asynchronous Clear and Preset and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>FJKCPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FJKCPE is a single J-K-type flip-flop with J, K, asynchronous clear (CLR), asynchronous preset (PRE), and clock enable (CE) inputs and data output (Q). When the asynchronous preset (PRE) is High and CLR is Low, all other inputs are ignored and Q is set High. The asynchronous preset (PRE), when High, and CLR set to Low overrides all other inputs and sets the Q output High. When CLR and PRE are Low and CE is High, Q responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock transition. Clock transitions are ignored when CE is Low.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FJKCPE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of fjkcpe is
begin
process (C, CLR, PRE)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (PRE='1') then
    Q <= '1';
  elsif (C'event and C='1') then
    if (CE='1') then
      if (J='0') then
        if (K='1') then
          Q <= '0';
        else
          Q <= not Q;
        end if;
      else
        if (K='0') then
          Q <= '1';
        else
          Q <= not Q;
        end if;
      end if;
    end if;
  end if;
end process;
end Behavioral;
```
Verilog Inference Code

```verilog
always @(posedge C or posedge CLR or posedge PRE)
begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (CE)
    begin
      if (!J)
        begin
          if (K)
            Q <= 0;
          end
        end
    else
      begin
        if (!K)
          Q <= 1;
        else
          Q <= !Q;
      end
end
```
FJKP

J-K Flip-Flop with Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FJKP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FJKP is a single J-K-type flip-flop with J, K, and asynchronous preset (PRE) inputs and data output (Q). The asynchronous preset (PRE) input, when High, overrides all other inputs and sets the Q output High. When PRE is Low, the Q output responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock transition.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>J</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
FJKP Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

FJKP Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FJKP Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```
architecture Behavioral of fjkp is

begin

process (C, PRE)
begin

if (PRE='1') then
    Q <= '1';
elsif (C'event and C='1') then
    if (J='0') then
        if (K='1') then
            Q <= '0';
        end if;
    else
        if (K='0') then
            Q <= '1';
        else
            Q <= not Q;
        end if;
    end if;
end if;
end process;
end Behavioral;
```

Verilog Inference Code

```
always @(posedge C or posedge PRE)
begin

if (PRE)
    Q <= 1;
else
begin

if (!J)
begin

if (K)
begin
    Q <= 0;
end
else
begin

if (!K)
begin
    Q <= 1;
else
    Q <= !Q;
end
end

end
end
```
FJKPE

J-K Flip-Flop with Clock Enable and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FJKPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FJKPE is a single J-K-type flip-flop with J, K, clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous preset (PRE), when High, overrides all other inputs and sets the Q output High. When PRE is Low and CE is High, the Q output responds to the state of the J and K inputs, as shown in the truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
FJKPE Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

FJKPE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
**Usage**

For HDL, this design element is inferred rather than instantiated.

**VHDL Inference Code**

```vhdl
architecture Behavioral of fjkpe is
begin
  process (C, PRE)
  begin
    if (PRE='1') then
      Q <= '1';
    elsif (C'event and C='1') then
      if (CE='1') then
        if (J='0') then
          if (K='1') then
            Q <= '0';
          end if;
        else
          if (K='0') then
            Q <= '1';
          else
            Q <= not Q;
          end if;
        end if;
      end if;
    end if;
  end process;
end Behavioral;
```

**Verilog Inference Code**

```verilog
always @(posedge C or posedge PRE)
begin
  if (PRE)
    Q <= 1;
  else if (CE)
```
begin
  if (!J)
    begin
      if (K)
        Q <= 0;
      end
    else
      begin
        if (!K)
          Q <= 1;
        else
          Q <= !Q;
      end
  end
end
J-K Flip-Flop with Clock Enable and Synchronous Reset and Set

FJKRSE is a single J-K-type flip-flop with J, K, synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). When synchronous reset (R) is High, all other inputs are ignored and output Q is reset Low. (Reset has precedence over Set.) When synchronous set (S) is High and R is Low, output Q is set High. When R and S are Low and CE is High, output Q responds to the state of the J and K inputs, according to the following truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FJKRSE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

FJKRSE Implementation Spartan-3, Virtex-II, Virtex-II Pro

FJKRSE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of fjkrse is
begin
process (C)
begin
if (C'event and C='1') then
if (R='1') then
Q <= '0';
elsif (S='1') then
Q <= '1';
elsif (CE='1') then
if (J='0') then
if (K='1') then
Q <= '0';
else
end if;
else
if (K='0') then
Q <= '1';
else
Q <= not Q;
end if;
end if;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
if (R)
Q <= 0;
else if (S)
Q <= 1;
else if (CE)
begin
if (!J)
begin
if (K)
Q <= 0;
end
else
begin
if (!K)
Q <= 1;
else
Q <= !Q;
end
end
end
FJKSRE

J-K Flip-Flop with Clock Enable and Synchronous Set and Reset

**Architectures Supported**

<table>
<thead>
<tr>
<th>FJKSRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FJKSRE is a single J-K-type flip-flop with J, K, synchronous set (S), synchronous reset (R), and clock enable (CE) inputs and data output (Q). When synchronous set (S) is High, all other inputs are ignored and output Q is set High. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, output Q is reset Low. When S and R are Low and CE is High, output Q responds to the state of the J and K inputs, as shown in the following truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FJKSRE will set when GSR is active. For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FJKSRE Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

FJKSRE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FJKSRE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of fjksre is
begin
process (C)
begin
if (C'event and C='1') then
if (S='1') then
Q <= '1';
elif (R='1') then
Q <= '0';
elif (CE='1') then
if (J='0') then
if (K='1') then
Q <= '0';
else
Q <= not Q;
end if;
elif (K='0') then
Q <= '1';
else
Q <= not Q;
end if;
else
if (K='0') then
Q <= '1';
else
Q <= not Q;
end if;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
if (S)
Q <= 1;
else if (R)
Q <= 0;
else if (CE)
begin
if (!J)
begin
if (K)
Q <= 0;
end
else
begin
if (!K)
Q <= 1;
else
Q <= !Q;
end
end
end
FMAP

F Function Generator Partitioning Control Symbol

Architectures Supported

<table>
<thead>
<tr>
<th>FMAP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The FMAP symbol is used to map logic to the function generator of a slice. See the appropriate CAE tool interface user guide for information about specifying this attribute in your schematic design editor.

The MAP=type parameter can be used with the FMAP symbol to further define how much latitude you want to give the mapping program. The following table shows MAP option characters and their meanings.

<table>
<thead>
<tr>
<th>MAP Option Character</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Pins.</td>
</tr>
<tr>
<td>C</td>
<td>Closed — Adding logic to or removing logic from the CLB is not allowed.</td>
</tr>
<tr>
<td>L</td>
<td>Locked — Locking CLB pins.</td>
</tr>
<tr>
<td>O</td>
<td>Open — Adding logic to or removing logic from the CLB is allowed.</td>
</tr>
<tr>
<td>U</td>
<td>Unlocked — No locking on CLB pins.</td>
</tr>
</tbody>
</table>

Possible types of MAP parameters for FMAP are MAP=PUC, MAP=PLC, MAP=PLO, and MAP=PUO. The default parameter is PUO. If one of the “open” parameters is used (PLO or PUO), only the output signals must be specified.

**Note:** Currently, only PUC and PUO are observed. PLC and PLO are translated into PUC and PUO, respectively.

The FMAP symbol can be assigned to specific CLB locations using LOC attributes.

Usage

FMAPs are generally inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate FMAPs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.
VHDL Instantiation Template

-- Component Declaration for FMAP should be placed
-- after architecture statement but before begin keyword

component FMAP
    port (I1 : in STD_ULOGIC;
         I2 : in STD_ULOGIC;
         I3 : in STD_ULOGIC;
         I4 : in STD_ULOGIC;
         O : in STD_ULOGIC);
end component;

-- Component Attribute specification for FMAP
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for FMAP should be placed
-- in architecture after the begin keyword

FMAP_INSTANCE_NAME : FMAP
    port map (I1 => user_I1,
              I2 => user_I2,
              I3 => user_I3,
              I4 => user_I4,
              O => user_O);

Verilog Instantiation Template

FMAP FMAP_instance_name (.I1 (user_I1),
                        .I2 (user_I2),
                        .I3 (user_I3),
                        .I4 (user_I4),
                        .O (user_O));

Commonly Used Constraints

BEL
BLKNM
HBLKNM
HU_SET
LOC
MAP
U_SET
FTC

Toggle Flip-Flop with Toggle Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FTC</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARTAN-3</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td></td>
</tr>
</tbody>
</table>

FTC is a synchronous, resettable toggle flip-flop. The asynchronous clear (CLR) input, when High, overrides all other inputs and resets the data output (Q) Low. The Q output toggles, or changes state, when the toggle enable (T) input is High and CLR is Low during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

FTC Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
FTC Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTC Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element can be instantiated or inferred.

VHDL Inference Code

architecture Behavioral of ftc is
begin
process (C, CLR)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (C'event and C='1') then
    if (T='1') then
      Q <= not Q;
    end if;
  end if;
end process;
end Behavioral;

Verilog Inference Code
always @(posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (T)
    Q <= !Q;
end
VHDL Instantiation Template

-- Component Declaration for FTC should be placed  
-- after architecture statement but before begin keyword

component FTC
    port (Q : out STD_ULOGIC;
       C : in STD_ULOGIC;
       CLR : in STD_ULOGIC;
       T : in STD_ULOGIC);
end component;

-- Component Attribute specification for FTC  
-- should be placed after architecture declaration but  
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for FTC should be placed  
-- in architecture after the begin keyword

FTC_INSTANCE_NAME : FTC
    port map (Q => user_Q,
              C => user_C,
              CLR => user_CLR,
              T => user_T);

Verilog Instantiation Template

FTC FTC_instance_name (.Q (user_Q),  
                        .C (user_C),  
                        .CLR (user_CLR),  
                        .T (user_T));

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, XBLKNM
FTCE

Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FTCE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTCE is a toggle flip-flop with toggle and clock enable and asynchronous clear. When the asynchronous clear (CLR) input is High, all other inputs are ignored and the data output (Q) is reset Low. When CLR is Low and toggle enable (T) and clock enable (CE) are High, Q output toggles, or changes state, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

```
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

FTCE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
FTCE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage
For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

```
architecture Behavioral of ftce is
begin
  process (C, CLR)
  begin
    if (CLR='1') then
      Q <= '0';
    elsif (C'event and C='1') then
      if (CE='1') then
        if (T='1') then
          Q <= not Q;
        end if;
      end if;
    end if;
  end process;
  Q <= Q;
end Behavioral;
```
Verilog Inference Code

```verilog
always @(posedge C or posedge CLR)
begin
    if (CLR)
        Q <= 0;
    else if (CE)
        if (T)
            Q <= !Q;
end
```

VHDL Instantiation Template

```vhdl
-- Component Declaration for FTCE should be placed
-- after architecture statement but before begin keyword

component FTCE
    port (Q : out STD_ULOGIC;
         C : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         CLR : in STD_ULOGIC;
         PRE : in STD_ULOGIC;
         T : in STD_ULOGIC);
end component;

-- Component Attribute specification for FTCE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for FTCE should be placed
-- in architecture after the begin keyword

FTCE_INSTANCE_NAME : FTCE
    port map (Q => user_Q,
              C => user_C,
              CE => user_CE,
              CLR => user_CLR,
              PRE => user_PRE,
              T => user_T);
```

Verilog Instantiation Template

```verilog
FTCE FTCE_instance_name (.Q (user_Q),
                        .C (user_C),
                        .CE (user_CE),
                        .CLR (user_CLR),
                        .PRE (user_PRE),
                        .T (user_T));
```

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, IOB, LOC, REG, RLOC, TIMEGRP, TNM, U_SET, and XBLKNM.
FTCLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FTCLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FTCLE is a toggle/loadable flip-flop with toggle and clock enable and asynchronous clear. When the asynchronous clear input (CLR) is High, all other inputs are ignored and output Q is reset Low. When load enable input (L) is High and CLR is Low, clock enable (CE) is overridden and the data on data input (D) is loaded into the flip-flop during the Low-to-High clock (C) transition. When toggle enable (T) and CE are High and L and CLR are Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTCLE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

FTCLE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTCLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of ftcle is
begin
  process (C, CLR)
  begin
    if (CLR='1') then
      Q <= '0';
    elsif (C'event and C='1') then
      if (L='1') then
        Q <= D;
      elsif (CE='1') then
        if (T='1') then
          Q <= not Q;
        end if;
      end if;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= !Q;
  end
end
```
FTCLEX

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FTCLEX</th>
</tr>
</thead>
</table>
| Spartan-II, Spartan-IIIE | Macro  
| Spartan-3 | Macro  
| Virtex, Virtex-E | Macro  
| Virtex-II, Virtex-II Pro, Virtex-II Pro X | Macro  
| XC9500, XC9500XV, XC9500X | Macro  
| CoolRunner XPLA3 | Macro  
| CoolRunner-II | Macro  

FTCLEX is a toggle/loadable flip-flop with toggle and clock enable and asynchronous clear. When the asynchronous clear input (CLR) is High, all other inputs are ignored and output Q is reset Low. When load enable input (L) is High, CLR is Low, and CE is High, the data on data input (D) is loaded into the flip-flop during the Low-to-High clock (C) transition. When toggle enable (T) and CE are High and L and CLR are Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTCLEX Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftclex is
begin
process (C, CLR)
begin
if (CLR='1') then
Q <= '0';
elsif (C'event and C='1') then
if (CE='1') then
if (L='1') then
Q <= D;
elsif (T='1') then
Q <= not Q;
end if;
end if;
end if;
end process;
end Behavioral;
Verilog Inference Code

always @(posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= !Q;
end
FTCP

Toggle Flip-Flop with Toggle Enable and Asynchronous Clear and Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FTCP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FTCP is a toggle flip-flop with toggle enable and asynchronous clear and preset. When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) is High and CLR is Low, all other inputs are ignored and Q is set High. When the toggle enable input (T) is High and CLR and PRE are Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of ftcp is
begin
process (C, CLR, PRE)
begin
    if (CLR='1') then
        Q <= '0';
    elsif (PRE='1') then
        Q <= '1';
    elsif (C'event and C='1') then
```
if (T='1') then
  Q <= not Q;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or posedge CLR or posedge PRE)
beg
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (T)
    Q <= !Q;
end

VHDL Instantiation Template

-- Component Declaration for FTCP should be placed
-- after architecture statement but before begin keyword

component FTCP
port (Q : out STD_ULOGIC;
  C : in STD_ULOGIC;
  CLR : in STD_ULOGIC;
  PRE : in STD_ULOGIC;
  T : in STD_ULOGIC);
end component;

-- Component Attribute specification for FTCP
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for FTCP should be placed
-- in architecture after the begin keyword

FTCP_INSTANCE_NAME : FTCP
port map (Q => user_Q,
  C => user_C,
  CLR => user_CLR,
  PRE => user_PRE,
  T => user_T);
Verilog Instantiation Template

FTCP FTCP_instance_name (.Q (user_Q),
   .C (user_C),
   .CLR (user_CLR),
   .PRE (user_PRE),
   .T (user_T));

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET
XBLKNM
FTCPE

Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear and Preset

**Architectures Supported**

<table>
<thead>
<tr>
<th>FTCPE</th>
<th>脊 modifications available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTCPE is a toggle flip-flop with toggle and clock enable and asynchronous clear and preset. When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) is High and CLR is Low, all other inputs are ignored and Q is set High. When the toggle enable input (T) and the clock enable input (CE) are High and CLR and PRE are Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition. Clock transitions are ignored when CE is Low.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FTCPE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

**Usage**

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of ftcpe is

begin

process (C, CLR, PRE)
begin

if (CLR='1') then
    Q <= '0';
elsif (PRE='1') then
    Q <= '1';
elif (C'event and C='1') then
    if (CE='1') then
        if (T='1') then
            Q <= not Q;
        end if;
    end if;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or posedge CLR or posedge PRE)
begin

    if (CLR)
        Q <= 0;
    else if (PRE)
        Q <= 1;
    else if (CE)
        if (T)
            Q <= '1Q;
        end
end
FTCPLE is a loadable toggle flip-flop with toggle and clock enable and asynchronous clear and preset. When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) is High and CLR is Low, all other inputs are ignored and Q is set High. When the load input (L) is High, the clock enable input (CE) is overridden and data on data input (D) is loaded into the flip-flop during the Low-to-High clock transition. When the toggle enable input (T) and the clock enable input (CE) are High and CLR, PRE, and L are Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition. Clock transitions are ignored when CE is Low.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.
FTCPLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftcple is

begin

process (C, CLR, PRE)
begin

if (CLR='1') then
    Q <= '0';
elsif (PRE='1') then
    Q <= '1';
elsif (C'event and C='1') then
    if (L='1') then
        Q <= D;
    elsif (CE='1') then
        if (T='1') then
            Q <= not Q;
        end if;
    end if;
end if;
end process;

end Behavioral;
Verilog Inference Code

always @(posedge C or posedge CLR or posedge PRE)
begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= !Q;
end
Dual Edge Triggered Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>FTDCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTDCE is a dual edge triggered toggle flip-flop with toggle and clock enable and asynchronous clear. When the asynchronous clear (CLR) input is High, all other inputs are ignored and the data output (Q) is reset Low. When CLR is Low and toggle enable (T) and clock enable (CE) are High, Q output toggles, or changes state, during the Low-to-High and High-to-Low clock (C) transitions. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

FTDCE Implementation CoolRunner-II
Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftdce is
begin
process (C, CLR)
begin
if (CLR='1') then
    Q <= '0';
elsif (C'event) then
    if (CE='1') then
        if (T='1') then
            Q <= not Q;
        end if;
    end if;
end if;
end process;
end Behavioral;

Verilog Inference Code
always @(posedge C or negedge C or posedge CLR)
begin
if (CLR)
    Q <= 0;
else if (CE)
    if (T)
        Q <= !Q;
end
FTDCLE

Dual Edge Triggered Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FTDCLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II Macro</td>
</tr>
</tbody>
</table>

FTDCLE is a dual edge triggered toggle/loadable flip-flop with toggle and clock enable and asynchronous clear. When the asynchronous clear input (CLR) is High, all other inputs are ignored and output Q is reset Low. When load enable input (L) is High and CLR is Low, clock enable (CE) is overridden and the data on data input (D) is loaded into the flip-flop during the Low-to-High and High-to-Low clock (C) transitions. When toggle enable (T) and CE are High and L and CLR are Low, output Q toggles, or changes state, during the Low- to-High and High-to-Low clock transitions. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTDCLE Implementation CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of ftdcle is
begin

  process (C, CLR)
  begin
    if (CLR='1') then
      Q <= '0';
    elsif (C'event) then
      if (L='1') then
        Q <= D;
      elsif (CE='1') then
        if (T='1') then
          Q <= not Q;
        end if;
      end if;
    end if;
  end process;

end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
```
else if (CE)
    if (T)
        Q <= !Q;
    end

FTDCLEX

Dual Edge Triggered Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>FTCLEX</th>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTDCLEX is a dual edge triggered toggle/loadable flip-flop with toggle and clock enable and asynchronous clear. When the asynchronous clear input (CLR) is High, all other inputs are ignored and output Q is reset Low. When load enable input (L) is High, CLR is Low, and CE is High, the data on data input (D) is loaded into the flip-flop during the Low-to-High and High-to-Low clock (C) transitions. When toggle enable (T) and CE are High and L and CLR are Low, output Q toggles, or changes state, during the Low-to-High and High-to-Low clock transitions. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR L CE T D C Q</td>
<td></td>
</tr>
<tr>
<td>1 X X X X X 0</td>
<td></td>
</tr>
<tr>
<td>0 1 1 X 1 ↑ 1</td>
<td></td>
</tr>
<tr>
<td>0 1 1 X 1 ↓ 1</td>
<td></td>
</tr>
<tr>
<td>0 1 1 X 1 ↑ 0</td>
<td></td>
</tr>
<tr>
<td>0 1 1 X 1 ↓ 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 X X X No Chg</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 X X No Chg</td>
<td></td>
</tr>
<tr>
<td>0 0 1 1 X ↑ Toggle</td>
<td></td>
</tr>
<tr>
<td>0 0 1 1 X ↓ Toggle</td>
<td></td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of ftdclex is

begin

process (C, CLR)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (C'event) then
    if (CE='1') then
      if (L='1') then
        Q <= D;
      elsif (T='1') then
        Q <= not Q;
      end if;
    end if;
  end if;
end process;

end Behavioral;

Verilog Inference Code

always @(posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= 'Q;
  end
end
FTDCP

Toggle Flip-Flop with Toggle Enable and Asynchronous Clear and Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FTDCP</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
<th>XC9500, XC9500XV, XC9500XL</th>
<th>CoolRunner XPLA3</th>
<th>CoolRunner-II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive</td>
<td>Primitive</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

FTDCP is a toggle flip-flop with toggle enable and asynchronous clear and preset. When the asynchronous clear (CLR) input is High, all other inputs are ignored and the output (Q) is reset Low. When the asynchronous preset (PRE) is High and CLR is Low, all other inputs are ignored and Q is set High. When the toggle enable input (T) is High and CLR and PRE are Low, output Q toggles, or changes state, during the Low-to-High and High-to-Low clock (C) transition.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTDCP Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code
architecture Behavioral of ftdcp is
begin
process (C, CLR, PRE)
begin
if (CLR='1') then
    Q <= '0';
elsif (PRE='1') then
    Q <= '1';
elsif (C'event) then
    if (T='1') then
        Q <= not Q;
    end if;
end if;
end process;
end Behavioral;

Verilog Inference Code
always @ (posedge C or negedge C or posedge CLR or posedge PRE)
begin
    if (CLR)
        Q <= 0;
    else if (PRE)
        Q <= 1;
    else if (T)
        Q <= !Q;
end
FTDRSE

Dual Edge Triggered Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set

Architectures Supported

<table>
<thead>
<tr>
<th>FTDRSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTDRSE is a dual edge triggered toggle flip-flop with toggle and clock enable and synchronous reset and set. When the synchronous reset input (R) is High, it overrides all other inputs and the data output (Q) is reset Low. When the synchronous set input (S) is High and R is Low, clock enable input (CE) is overridden and output Q is set High. (Reset has precedence over Set.) When toggle enable input (T) and CE are High and R and S are Low, output Q toggles, or changes state, during the Low-to-High and High-to-Low clock transitions.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.
FTDRSE Implementation CoolRunner-II

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of ftdrse is
begin
    process (C)
    begin
        if (C'event) then
            if (R='1') then
                Q <= '0';
            elsif (S='1') then
                Q <= '1';
            elsif (CE='1') then
                if (T='1') then
                    Q <= not Q;
                end if;
            end if;
        end if;
    end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or negedge C)
begin
    if (R)
        Q <= 0;
    else if (S)
        Q <= 1;
    else if (CE)
```
if (T)  
Q <= !Q;
end
FTDRSLE

Dual Edge Triggered Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set

Architectures Supported

<table>
<thead>
<tr>
<th>FTDRSLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FTDRSLE is a dual edge triggered toggle/loadable flip-flop with toggle and clock enable and synchronous reset and set. The synchronous reset input (R), when High, overrides all other inputs and resets the data output (Q) Low. (Reset has precedence over Set.) When R is Low and synchronous set input (S) is High, the clock enable input (CE) is overridden and output Q is set High. When R and S are Low and load enable input (L) is High, CE is overridden and data on data input (D) is loaded into the flip-flop during the Low-to-High and High-to-Low clock transitions. When R, S, and L are Low and CE is High, output Q toggles, or changes state, during the Low-to-High and High-to-Low clock transitions. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTDRSLE Implementation CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftdrsle is

begin

process (C)
begin

if (C'event) then

if (R='1') then
Q <= '0';
elsif (S='1') then
Q <= '1';
elsif (L='1') then
Q <= D;
elsif (CE='1') then
if (T='1') then
Q <= not Q;
end if;
end if;
end if;
end process;
end Behavioral;
Verilog Inference Code

always @(posedge C or negedge C)
begin
  if (R)
    Q <= 0;
  else if (S)
    Q <= 1;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= !Q;
end
FTP

Toggle Flip-Flop with Toggle Enable and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>FTP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTP is a toggle flip-flop with toggle enable and asynchronous preset. When the asynchronous preset (PRE) input is High, all other inputs are ignored and output Q is set High. When toggle-enable input (T) is High and PRE is Low, output Q toggles, or changes state, during the Low-to-High clock (C) transition.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset to output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_VIRTEX, STARTUP_SPARTAN3, or the STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

FTP Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
FTP Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTP Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

architecture Behavioral of ftp is
begin
process (C, PRE)
begin
if (PRE='1') then
Q <= '1';
elsif (C'event and C='1') then
if (T='1') then
Q <= not Q;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C or posedge PRE)
begin
if (PRE)
Q <= 1;
else if (T)
Q <= !Q;
end
VHDL Instantiation Template

-- Component Declaration for FTP should be placed
-- after architecture statement but before begin keyword

component FTP
    port (Q : out STD_ULOGIC;
         C : in STD_ULOGIC;
         PRE : in STD_ULOGIC;
         T : in STD_ULOGIC);
end component;

-- Component Attribute specification for FTP
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for FTP should be placed
-- in architecture after the begin keyword

FTP_INSTANCE_NAME : FTP
    port map (Q => user_Q,
             C => user_C,
             PRE => user_PRE,
             T => user_T);

Verilog Instantiation Template

FTP FTP_instance_name (.Q (user_Q),
                      .C (user_C),
                      .PRE (user_PRE),
                      .T (user_T));

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
IOB
LOC
REG
RLOC
TIMEGRP
TNM
U_SET, XBLKNM
FTPE

Toggle Flip-Flop with Toggle and Clock Enable and Asynchronous Preset

Architecture Supported

<table>
<thead>
<tr>
<th>FTPE</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td></td>
</tr>
</tbody>
</table>

FTPE is a toggle flip-flop with toggle and clock enable and asynchronous preset. When the asynchronous preset (PRE) input is High, all other inputs are ignored and output Q is set High. When the toggle enable input (T) is High, clock enable (CE) is High, and PRE is Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset to output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of ftpe is

begin

process (C, PRE)
begin

if (PRE='1') then
    Q <= '1';
elsif (C'event and C='1') then
    if (CE='1') then
        if (T='1') then
            Q <= not Q;
        end if;
    end if;
end if;
end process;

end Behavioral;
```
Verilog Inference Code

always @(posedge C or posedge PRE)
begin
  if (PRE)
    Q <= 1;
  else if (CE)
    if (T)
      Q <= !Q;
end
FTPLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>Architectures Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

FTPLE is a toggle/loadable flip-flop with toggle and clock enable and asynchronous preset. When the asynchronous preset input (PRE) is High, all other inputs are ignored and output Q is set High. When the load enable input (L) is High and PRE is Low, the clock enable (CE) is overridden and the data (D) is loaded into the flip-flop during the Low-to-High clock transition. When L and PRE are Low and toggle-enable input (T) and CE are High, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously cleared, output Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously preset to output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTPLE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

FTPLE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTPLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftple is

begin

process (C, PRE)
begin
  if (PRE='1') then
    Q <= '1';
  elsif (C'event and C='1') then
    if (L='1') then
      Q <= D;
    elsif (CE='1') then
      if (T='1') then
        Q <= not Q;
      end if;
    end if;
  end if;
end process;

end Behavioral;

Verilog Inference Code

always @(posedge C or posedge PRE)
begin
  if (PRE)
    Q <= 1;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= !Q;
  end
FTRSE

Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set

Architectures Supported

<table>
<thead>
<tr>
<th>FTRSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>Macro</td>
</tr>
</tbody>
</table>

FTRSE is a toggle flip-flop with toggle and clock enable and synchronous reset and set. When the synchronous reset input (R) is High, it overrides all other inputs and the data output (Q) is reset Low. When the synchronous set input (S) is High and R is Low, clock enable input (CE) is overridden and output Q is set High. (Reset has precedence over Set.) When toggle enable input (T) and CE are High and R and S are Low, output Q toggles, or changes state, during the Low-to-High clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTRSE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

FTRSE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTRSE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of ftrse is

begin

process (C)
begin

if (C'event and C='1') then

if (R='1') then

Q <= '0';

elsif (S='1') then

Q <= '1';

elsif (CE='1') then

if (T='1') then

Q <= not Q;

end if;

end if;

end if;
end process;

end Behavioral;

Verilog Inference Code

always @(posedge C)
begin

if (R)

Q <= 0;

else if (S)

Q <= 1;

else if (CE)

if (T)

Q <= !Q;

end
FTRSLE

Toggle/Loadable Flip-Flop with Toggle and Clock Enable and Synchronous Reset and Set

Architectures Supported

<table>
<thead>
<tr>
<th>FTRSLE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro,</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II Pro X</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTRSLE is a toggle/loadable flip-flop with toggle and clock enable and synchronous reset and set. The synchronous reset input (R), when High, overrides all other inputs and resets the data output (Q) Low. (Reset has precedence over Set.) When R is Low and synchronous set input (S) is High, the clock enable input (CE) is overridden and output Q is set High. When R and S are Low and load enable input (L) is High, CE is overridden and data on data input (D) is loaded into the flip-flop during the Low-to-High clock transition. When R, S, and L are Low, CE is High and T is High, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R S L CE T D C</td>
<td>Q</td>
</tr>
<tr>
<td>1 0 X X X X</td>
<td>↑ 0</td>
</tr>
<tr>
<td>0 1 X X X X</td>
<td>↑ 1</td>
</tr>
<tr>
<td>0 0 1 X X 1</td>
<td>↑ 1</td>
</tr>
<tr>
<td>0 0 1 X X 0</td>
<td>↑ 0</td>
</tr>
<tr>
<td>0 0 0 0 X X</td>
<td>No Chg</td>
</tr>
<tr>
<td>0 0 0 1 0 X</td>
<td>No Chg</td>
</tr>
<tr>
<td>0 0 0 1 1 X</td>
<td>↑ Toggle</td>
</tr>
</tbody>
</table>
FTRSLE Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

FTRSLE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTRSLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftrsle is
begin
process (C)
begin
if (C'event and C='1') then
  if (R='1') then
    Q <= '0';
  elsif (S='1') then
    Q <= '1';
  elsif (L='1') then
    Q <= D;
  elsif (CE='1') then
    if (T='1') then
      Q <= not Q;
    end if;
  end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
  if (R)
    Q <= 0;
  else if (S)
    Q <= 1;
  else if (L)
    Q <= D;
  else if (CE)
    if (T)
      Q <= !Q;
  end
FTSRE

Toggle Flip-Flop with Toggle and Clock Enable and Synchronous Set and Reset

Architectures Supported

<table>
<thead>
<tr>
<th>FTSRE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

FTSRE is a toggle flip-flop with toggle and clock enable and synchronous set and reset. The synchronous set input, when High, overrides all other inputs and sets data output (Q) High. (Set has precedence over Reset.) When synchronous reset input (R) is High and S is Low, clock enable input (CE) is overridden and output Q is reset Low. When toggle enable input (T) and CE are High and S and R are Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FTSRE will set when GSR is active. For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
FTSRE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

FTSRE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTSRE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of ftsre is
begin

process (C)
begin
if (C'event and C='1') then
  if (S='1') then
    Q <= '1';
  elsif (R='1') then
    Q <= '0';
  elsif (CE='1') then
    if (T='1') then
      Q <= not Q;
    end if;
  end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
if (S)
  Q <= 1;
else if (R)
  Q <= 0;
else if (CE)
  if (T)
    Q <= 'Q;
end
FTSRLE is a toggle/loadable flip-flop with toggle and clock enable and synchronous set and reset. The synchronous set input (S), when High, overrides all other inputs and sets data output (Q) High. (Set has precedence over Reset.) When synchronous reset (R) is High and S is Low, clock enable input (CE) is overridden and output Q is reset Low. When load enable input (L) is High and S and R are Low, CE is overridden and data on data input (D) is loaded into the flip-flop during the Low-to-High clock transition. When the toggle enable input (T) and CE are High and S, R, and L are Low, output Q toggles, or changes state, during the Low-to-High clock transition. When CE is Low, clock transitions are ignored.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the flip-flop is asynchronously preset when a High-level pulse is applied on the PRLD global net.

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is asynchronously cleared, output Low, when global set/reset (GSR) is active.

The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol. FTSRLE will set when GSR is active. For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the flip-flop is preset to active high when GSR is active.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S R L CE T D C</td>
<td>Q</td>
</tr>
<tr>
<td>1 X X X X X ↑</td>
<td>1</td>
</tr>
<tr>
<td>0 1 X X X X ↑</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1 X X 1 ↑</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1 X X 0 ↑</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 0 0 X X No Chg</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 1 0 X X No Chg</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 X ↑ Toggle</td>
<td></td>
</tr>
</tbody>
</table>
FTSRLE Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

FTSRLE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

FTSRLE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of ftsrle is
begin
process (C)
begin
if (C'event and C='1') then
if (S='1') then
Q <= '1';
elsif (R='1') then
Q <= '0';
elsif (L='1') then
Q <= D;
elsif (CE='1') then
if (T='1') then
Q <= not Q;
end if;
end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(posedge C)
begin
if (S)
Q <= 1;
else if (R)
Q <= 0;
else if (L)
Q <= D;
else if (CE)
if (T)
Q <= !Q;
end
**GND**

**Ground-Connection Signal Tag**

**Architectures Supported**

<table>
<thead>
<tr>
<th>GND</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

The GND signal tag, or parameter, forces a net or input function to a Low logic level. A net tied to GND cannot have any other source.

When the logic-trimming software or fitter encounters a net or input function tied to GND, it removes any logic that is disabled by the GND signal. The GND signal is only implemented when the disabled logic cannot be removed.

**Usage**

For HDL, this design element can be instantiated or inferred.

**VHDL Inference Code:**

```vhdl
gnd_signal <= '0';
```

**Verilog Inference Code:**

```verilog
assign gnd_signal = 0;
```

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for GND should be placed
-- after architecture statement but before begin keyword

component GND
    port (G : out STD_ULOGIC);
end component;

-- Component Attribute specification for GND
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GND should be placed
-- in architecture after the begin keyword
```
Verilog Instantiation Template

GND GND_instance_name (.G (user_G));

Commonly Used Constraints
None
**GT_AURORA_n**

Gigabit Transceiver for High-Speed I/O

**Architectures Supported**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
<tr>
<td><em>Supported for Virtex-II Pro but not for Virtex-II or Virtex-II Pro X.</em></td>
<td></td>
</tr>
</tbody>
</table>

This Xilinx protocol gigabit transceiver supports 1, 2, and 4-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the *RocketIO Transceiver User Guide* for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the *RocketIO Transceiver User Guide.*
GT_AURORA_1

- BREFCLK
- BREFCLK2
- CHBOND(3:0)
- CONFIGENABLE
- CONFIGIN
- ENCHANSYNC
- ENMCOMMAALIGN
- ENPCOMMAALIGN
- LOOPBACK(1:0)
- POWERDOWN
- REFCLK
- REFCLK2
- REFCLKSEL
- RXN
- RXP
- RXPOLARITY
- RXRESET
- RXUSRCLK
- RXUSRCLK2
- TXBYPASS8B10B(0:0)
- TXCHARDISPMODE(0:0)
- TXCHARDISPV(0:0)
- TXCHARISK(0:0)
- TXDATA(7:0)
- TXFORCECRCERR
- TXINHIBIT
- TXPOLARITY
- TXRESET
- TXUSRCLK
- TXUSRCLK2

CHBONDDONE
CHBONDO(3:0)
CONFIGOUT
RXBUFSTATUS(1:0)
RXCHARISCOMMA(0:0)
RXCHARISK(0:0)
RXCHECKINGCRC
RXCLKCORCNT(2:0)
RXCOMMADET
RXCRCERR
RXDATA(7:0)
RXDISPERR(0:0)
RXLOSSOFSYNC(1:0)
RXNOTINTABLE(0:0)
RXREALIGN
RXRECCLK
RXRUNDISP(0:0)
TXBUFERR
TXKERR(0:0)
TXN
TXP
TXRUNDISP(0:0)
### GT_AURORA_2

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>CHBONO(3:0)</td>
</tr>
<tr>
<td>CHBONDI(3:0)</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>RXCHARISK(1:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(15:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(1:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(1:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(1:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(1:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPMODEVAL(1:0)</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

**GT_AURORA_1 VHDL Code**

```vhdl
-- Component Attribute specification for GT_AURORA_1
-- should be placed after architecture declaration but
-- before the begin keyword
```
component GT_AURORA_1
-- synopsys translate_off
generic (  
  ALIGN_COMMA_MSB          : boolean := FALSE;
  CHAN_BOND_LIMIT          : integer := 16;
  CHAN_BOND_MODE           : string := "OFF";
  CHAN_BOND_OFFSET         : integer := 8;
  CHAN_BOND_ONE_SHOT       : boolean := FALSE;
  CHAN_BOND_SEQ_1_1        : bit_vector := "00101111100";
  CHAN_BOND_SEQ_1_2        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_3        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_4        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_1        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_2        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_3        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_4        : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_USE      : boolean := FALSE;
  CHAN_BOND_SEQ_LEN        : integer := 1;
  CHAN_BOND_WAIT           : integer := 8;
  CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
  CLK_COR_KEEP_IDLE        : boolean := FALSE;
  CLK_COR_REPEAT_WAIT      : integer := 1;
  CLK_COR_SEQ_1_1          : bit_vector := "00111110111";
  CLK_COR_SEQ_1_2          : bit_vector := "00000000000";
  CLK_COR_SEQ_1_3          : bit_vector := "00000000000";
  CLK_COR_SEQ_1_4          : bit_vector := "00000000000";
  CLK_COR_SEQ_2_1          : bit_vector := "00000000000";
  CLK_COR_SEQ_2_2          : bit_vector := "00000000000";
  CLK_COR_SEQ_2_3          : bit_vector := "00000000000";
  CLK_COR_SEQ_2_4          : bit_vector := "00000000000";
  CLK_COR_SEQ_2_USE        : boolean := FALSE;
  CLK_COR_SEQ_LEN          : integer := 1;
  CLK_COR_SEQ_LEN          : integer := 1;
  CLK_COR_SEQ_LEN          : integer := 1;
  COMMA_10B_MASK           : bit_vector := "1111111111";
  CRC_END_OF_PKT           : string := "K29_7";
  CRC_FORMAT               : string := "USER_MODE";
  CRC_START_OF_PKT         : string := "K27_7";
  DEC_MCOMMA_DETECT        : boolean := TRUE;
  DEC_PCOMMA_DETECT        : boolean := TRUE;
  DEC_VALID_COMMA_ONLY     : boolean := TRUE;
  MCOMMA_10B_VALUE         : bit_vector := "1100000101";
  MCOMMA_DETECT            : boolean := TRUE;
  PCOMMA_10B_VALUE         : bit_vector := "0011111010";
  PCOMMA_DETECT            : boolean := TRUE;
  REF_CLK_V_SEL            : integer := 0;
  RX_BUFFER_USE            : boolean := TRUE;
  RX_CRC_USE               : boolean := FALSE;
  RX_DATA_WIDTH            : integer := 1;
  RX_DECODE_USE            : boolean := TRUE;
  RX_LOS_INVALID_INCR      : integer := 1;
  RX_LOS_THRESHOLD         : integer := 4;
  RXLOSS_OF_SYNC_FSM       : boolean := TRUE;
  SERDES_10B               : boolean := FALSE;
  TERMINATION_IMP          : integer := 50;
  TX_BUFFER_USE            : boolean := TRUE;
  TX_CRC_FORCE_VALUE       : bit_vector := "11010110";
  TX_CRC_USE               : boolean := FALSE;
  TX_DATA_WIDTH            : integer := 1;  
)
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port (  
  CHBONDDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(3 downto 0);
  CONFIGOUT : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(0 downto 0);
  RXCHARISK : out std_logic_vector(0 downto 0);
  RXCHECKINGCRC : out std_ulogic;
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_logic_vector(0 downto 0);
  RXCRCERR : out std_ulogic;
  RXREALIGN : out std_ulogic;
  RXDATA : out std_logic_vector(7 downto 0);
  RXDISPERR : out std_logic_vector(0 downto 0);
  RXDISP : out std_logic_vector(0 downto 0);
  RXN : out std_ulogic;
  RXPOLARITY : out std_ulogic;
  RXRESET : out std_ulogic;
  RXUSRCLK : in std_ulogic;
  RXUSRCLK2 : in std_ulogic;
  BREFCLK : in std_ulogic := 'X';
  BREFCLK2 : in std_ulogic := 'X';
  CHBONDI : in std_logic_vector(3 downto 0);
  CONFIGENABLE : in std_ulogic;
  CONFIGIN : in std_ulogic;
  ENCHANSYNC : in std_ulogic;
  ENMCOMMAALIGN : in std_ulogic;
  LOOPBACK : in std_logic_vector(1 downto 0);
  POWERDOWN : in std_ulogic;
  REFCLK : in std_ulogic;
  REFCLK2 : in std_ulogic;
  REFCKSEL : in std_ulogic;
  RXN : in std_ulogic;
  RXP : in std_ulogic;
  RXPOLARITY : in std_ulogic;
  RXRESET : in std_ulogic;
  RXUSRCLK : in std_ulogic;
  RXUSRCLK2 : in std_ulogic;
  TXBYPASS8B10B : in std_logic_vector(0 downto 0);
  TXCHARDISPMODE : in std_logic_vector(0 downto 0);
  TXCHARDISPVAL : in std_logic_vector(0 downto 0);
  TXCHARISK : in std_logic_vector(0 downto 0);
  TXDATA : in std_logic_vector(7 downto 0);
  TXFORCECRCERR : in std_ulogic;
  TXINHIBIT : in std_ulogic;
  TXPOLARITY : in std_ulogic;
  TXRESET : in std_ulogic;
  TXUSRCLK : in std_ulogic;
  TXUSRCLK2 : in std_ulogic);
end component;

-- Component Attribute specification for GT_AURORA_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_AURORA_1 should be placed
-- in architecture after the begin keyword

GT_AURORA_1_INSTANCE_NAME : GT_AURORA_1
-- synopsys translate_off
generic map ( 
  ALIGN_COMMA_MSB => boolean_value,
  CHAN_BOND_LIMIT => integer_value,
  CHAN_BOND_MODE => string_value,
  CHAN_BOND_OFFSET => integer_value,
  CHAN_BOND_ONE_SHOT => boolean_value,
  CHAN_BOND_SEQ_1_1 => bit_value,
  CHAN_BOND_SEQ_1_2 => bit_value,
  CHAN_BOND_SEQ_1_3 => bit_value,
  CHAN_BOND_SEQ_1_4 => bit_value,
  CHAN_BOND_SEQ_2_1 => bit_value,
  CHAN_BOND_SEQ_2_2 => bit_value,
  CHAN_BOND_SEQ_2_3 => bit_value,
  CHAN_BOND_SEQ_2_4 => bit_value,
  CHAN_BOND_SEQ_2_USE => boolean_value,
  CHAN_BOND_SEQ_LEN => integer_value,
  CHAN_BOND_WAIT => integer_value,
  CLK_COR_INSERT_IDLE_FLAG => boolean_value,
  CLK_COR_KEEP_IDLE => boolean_value,
  CLK_COR_REPEAT_WAIT => integer_value,
  CLK_COR_SEQ_1_1 => bit_value,
  CLK_COR_SEQ_1_2 => bit_value,
  CLK_COR_SEQ_1_3 => bit_value,
  CLK_COR_SEQ_1_4 => bit_value,
  CLK_COR_SEQ_2_1 => bit_value,
  CLK_COR_SEQ_2_2 => bit_value,
  CLK_COR_SEQ_2_3 => bit_value,
  CLK_COR_SEQ_2_4 => bit_value,
  CLK_COR_SEQ_2_USE => boolean_value,
  CLK_COR_SEQ_LEN => integer_value,
  CLK_CORRECT_USE => boolean_value,
  COMMA_10B_MASK => bit_value,
  CRC_END_OF_PKT => string_value,
  CRC_FORMAT => string_value,
  CRC_START_OF_PKT => string_value,
  DEC_MCOMMA_DETECT => boolean_value,
  DEC_PCOMMA_DETECT => boolean_value,
  DEC_VALID_COMMA_ONLY => boolean_value,
  MCOMMA_10B_VALUE => bit_value,
  MCOMMA_DETECT => boolean_value,
  PCOMMA_10B_VALUE => bit_value,
  PCOMMA_DETECT => boolean_value,
  REF_CLK_V_SEL => integer_value,
  RX_BUFFER_USE => boolean_value,
  RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (CHBONDDONE => user_CHBONDDONE,
CHBONO => user_CHBONO,
CONFIGOUT => user_CONFIGOUT,
RXBUFFSTATUS => user_RXBUFFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CHBOND1 => user_CHBOND1,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE  => user_TXCHARDISPMODE,
TXCHARDISPVAL   => user_TXCHARDISPVAL,
TXCHARISK       => user_TXCHARISK,
TXDATA          => user_TXDATA,
TXFORCERCERR    => user_TXFORCERCERR,
TXINHIBIT       => user_TXINHIBIT,
TXPOLARITY      => user_TXPOLARITY,
TXRESET         => user_TXRESET,
TXUSRCLK        => user_TXUSRCLK,
TXUSRCLK2       => user_TXUSRCLK2);

GT_AURORA_2 VHDL Code
-- Component Attribute specification for GT_AURORA_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_AURORA_2
-- synopsys translate_off
generic (
  ALIGN_COMMA_MSB     : boolean := FALSE;
  CHAN_BOND_LIMIT     : integer := 16;
  CHAN_BOND_MODE      : string := "OFF";
  CHAN_BOND_OFFSET    : integer := 8;
  CHAN_BOND_ONE_SHOT  : boolean := FALSE;
  CHANNEL_1_1         : bit_vector := "00101111100";
  CHANNEL_1_2         : bit_vector := "00000000000";
  CHANNEL_1_3         : bit_vector := "00000000000";
  CHANNEL_1_4         : bit_vector := "00000000000";
  CHANNEL_2_1         : bit_vector := "00111110111";
  CHANNEL_2_2         : bit_vector := "00111110111";
  CHANNEL_2_3         : bit_vector := "00000000000";
  CHANNEL_2_4         : bit_vector := "00000000000";
  CHANNEL_2_USE       : boolean := FALSE;
  CHANNEL_SEQ_LEN     : integer := 1;
  CHANNEL_WAIT        : integer := 8;
  CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
  CLK_COR_KEEP_IDLE   : boolean := FALSE;
  CLK_COR_REPEAT_WAIT : integer := 1;
  CLK_COR_SEQ_1_1     : bit_vector := "00111110111";
  CLK_COR_SEQ_1_2     : bit_vector := "00111110111";
  CLK_COR_SEQ_1_3     : bit_vector := "00000000000";
  CLK_COR_SEQ_1_4     : bit_vector := "00000000000";
  CLK_COR_SEQ_2_1     : bit_vector := "00111110111";
  CLK_COR_SEQ_2_2     : bit_vector := "00000000000";
  CLK_COR_SEQ_2_3     : bit_vector := "00000000000";
  CLK_COR_SEQ_2_4     : bit_vector := "00000000000";
  CLK_COR_SEQ_2_USE   : boolean := FALSE;
  CLK_COR_SEQ_LEN     : integer := 2;
  CLK_CORRECT_USE     : boolean := TRUE;
  COMMA_10B_MASK      : bit_vector := "11111111111";
  CRC_END_OF_PKT      : string := "K29_7";
  CRC_FORMAT          : string := "USER_MODE";
  CRC_START_OF_PKT    : string := "K27_7";
  DEC_MCOMMA_DETECT   : boolean := TRUE;
  DEC_PCOMMA_DETECT   : boolean := TRUE;
  DEC_VALID_COMMA_ONLY : boolean := TRUE;
  MCOMMA_10B_VALUE    : bit_vector := "1100000101";

Libraries Guide
www.xilinx.com
ISE 6.li
1-800-255-7778
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "0011111010";
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 2;
RX_DECIMATE : boolean := TRUE;
RX LOS_INVALID_INCREMENT : integer := 1;
RX LOSS_THRESHOLD : integer := 4;
RX LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 2;
TX DIFF_CTRL : integer := 500;
TX PREEMPHASIS : integer := 0);

-- synopsys translate_on
port (
CHBONDDONE : out std_ulogic;
CHBONO : out std_logic_vector(3 downto 0);
CONFIGOUT : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RCHARISCOMMA : out std_logic_vector(1 downto 0);
RCHARISK : out std_logic_vector(1 downto 0);
R CHECKINGCRC : out std_ulogic;
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADECT : out std_ulogic;
RXCRCERR : out std_ulogic;
RXDATA : out std_logic_vector(15 downto 0);
RXDISPor : out std_logic_vector(1 downto 0);
RX LOSSOFSYNC : out std_logic_vector(1 downto 0);
RX NOTINTABLE : out std_logic_vector(1 downto 0);
RXREALIGN : out std_ulogic;
RX RECKL : out std_ulogic;
RX RUNDISP : out std_logic_vector(1 downto 0);
TXXFERR : out std_ulogic;
TXXERR : out std_logic_vector(1 downto 0);
TXN : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(1 downto 0);
BREFCLK : in std_ulogic := 'X';
BREFCLK2 : in std_ulogic := 'X';
CHBOND : in std_logic_vector(3 downto 0);
CONFIGENABLE : in std_ulogic;
CONFIGIN : in std_ulogic;
ENCHANSYNC : in std_ulogic;
ENMACOMMAALIGN : in std_ulogic;
ENPCMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXN : in std_ulogic;
end component;

-- Component Attribute specification for GT_AURORA_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_AURORA_2 should be placed
-- in architecture after the begin keyword

GT_AURORA_2_INSTANCE_NAME : GT_AURORA_2
-- synopsys translate_off
generic map (    ALIGN_COMMA_MSB           => boolean_value,
                 CHAN_BOND_LIMIT           => integer_value,
                 CHAN_BOND_MODE            => string_value,
                 CHAN_BOND_OFFSET          => integer_value,
                 CHAN_BOND_ONE_SHOT        => boolean_value,
                 CHAN_BOND_SEQ_1_1         => bit_value,
                 CHAN_BOND_SEQ_1_2         => bit_value,
                 CHAN_BOND_SEQ_1_3         => bit_value,
                 CHAN_BOND_SEQ_1_4         => bit_value,
                 CHAN_BOND_SEQ_2_1         => bit_value,
                 CHAN_BOND_SEQ_2_2         => bit_value,
                 CHAN_BOND_SEQ_2_3         => bit_value,
                 CHAN_BOND_SEQ_2_4         => bit_value,
                 CHAN_BOND_SEQ_2_USE       => boolean_value,
                 CHAN_BOND_SEQ_LEN         => integer_value,
                 CHAN_BOND_WAIT            => integer_value,
                 CLK_COR_INSERT_IDLE_FLAG => boolean_value,
                 CLK_COR_KEEP_IDLE        => boolean_value,
                 CLK_COR_REPEAT_WAIT      => integer_value,
                 CLK_COR_SEQ_1_1          => bit_value,
                 CLK_COR_SEQ_1_2          => bit_value,
                 CLK_COR_SEQ_1_3          => bit_value,
                 CLK_COR_SEQ_1_4          => bit_value,
                 CLK_COR_SEQ_2_1          => bit_value,
                 CLK_COR_SEQ_2_2          => bit_value,
                 CLK_COR_SEQ_2_3          => bit_value,
                 CLK_COR_SEQ_2_4          => bit_value,)
CLK_COR_SEQ_2_USE              => boolean_value,
CLK_COR_SEQ_LEN                 => integer_value,
CLK_CORRECT_USE                 => boolean_value,
COMMA_10B_MASK                  => bit_value,
CRC_END_OF_PKT                  => string_value,
CRC_FORMAT                      => string_value,
CRC_START_OF_PKT                => string_value,
DEC_MCOMMA_DETECT               => boolean_value,
DEC_PCOMMA_DETECT               => boolean_value,
DEC_VALID_COMMA_ONLY            => boolean_value,
MCOMMA_10B_VALUE                => bit_value,
MCOMMA_DETECT                   => boolean_value,
PCOMMA_10B_VALUE                => bit_value,
PCOMMA_DETECT                   => boolean_value,
REF_CLK_V_SEL                   => integer_value,
RX_BUFFER_USE                   => boolean_value,
RX_CRC_USE                      => boolean_value,
RX_DATA_WIDTH                   => integer_value,
RXDecode_USE                    => boolean_value,
RX_LOS_INVALID_INCR             => integer_value,
RX_LOS_THRESHOLD                => integer_value,
RX_LOSS_OF_SYNC_FSM             => boolean_value,
SERDES_10B                      => boolean_value,
TERMINATION_IMP                 => integer_value,
TX_BUFFER_USE                   => boolean_value,
TX_CRC_FORCE_VALUE              => bit_value,
TX_CRC_USE                      => boolean_value,
TX_DATA_WIDTH                   => integer_value,
TX_DIFF_CTRL                    => integer_value,
TX_PREEMPHASIS                  => integer_value)

-- synopsys translate_on
port map (}

CHBONDDONE                      => user_CHBONDDONE,
CHBONDO                         => user_CHBONDO,
CONFIGOUT                       => user_CONFIGOUT,
RXBUFSTATUS                     => user_RXBUFSTATUS,
RXCHARISCOMMA                   => user_RXCHARISCOMMA,
RXCHARISK                       => user_RXCHARISK,
RXCHECKINGCRC                   => user_RXCHECKINGCRC,
RXCLKCORCNT                     => user_RXCLKCORCNT,
RXCOMMADET                      => user_RXCOMMADET,
RXCRCERR                        => user_RXCRCERR,
RXDATA                          => user_RXDATA,
RXDISPERR                       => user_RXDISPERR,
RXLOSSOFSYNC                    => user_RXLOSSOFSYNC,
RXNOTINTABLE                    => user_RXNOTINTABLE,
RXREALIGN                       => user_RXREALIGN,
RXRECLK                        => user_RXRECLK,
RXRUNDISP                       => user_RXRUNDISP,
TXBUFERR                        => user_TXBUFERR,
TXKERR                          => user_TXKERR,
TXN                             => user_TXN,
TXP                             => user_TXP,
TXRUNDISP                       => user_TXRUNDISP,
BREFCLK                         => user_BREFCLK,
BREFCLK2                        => user_BREFCLK2,
CHBONDI                         => user_CHBONDI,
GT_AURORA_4 VHDL Code

-- Component Attribute specification for GT_AURORA_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_AURORA_4
-- synopsys translate_off
generic (
  ALIGN_COMMA_MSB              : boolean := FALSE;
  CHAN_BOND_LIMIT              : integer := 16;
  CHAN_BOND_MODE               : string := "OFF";
  CHAN_BOND_OFFSET             : integer := 8;
  CHAN_BOND_ONE_SHOT           : boolean := FALSE;
  CHAN_BOND_SEQ_1_1            : bit_vector := "00101111100";
  CHAN_BOND_SEQ_1_2            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_3            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_4            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_1            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_2            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_3            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_4            : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_USE          : boolean := FALSE;
  CHAN_BOND_SEQ_LEN            : integer := 1;
  CHAN_BOND_WAIT               : integer := 8;
  CLK_COR_INSERT_IDLE_FLAG     : boolean := FALSE;
  CLK_COR_KEEP_IDLE            : boolean := FALSE;
  CLK_COR_REPEAT_WAIT          : integer := 1;
  CLK_COR_SEQ_1_1              : bit_vector := "00111111011";
  CLK_COR_SEQ_1_2              : bit_vector := "00111111011";
);
CLK_COR_SEQ_1_3 : bit_vector := "00111110111";
CLK_COR_SEQ_1_4 : bit_vector := "00111110111";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 4;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "11111111111";
CRC_END_OF_PKT : string := "K29_7"
CRC_FORMAT : string := "USER_MODE"
CRC_START_OF_PKT : string := "K27_7"
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_Comma ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "11000001011"
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "00111110101"
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 4;
RXDecode_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "1100111010"
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 4;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on

port (}

CHBONDDONE : out std_ulogic;
CHBONDO : out std_logic_vector(3 downto 0);
CONFIGOUT : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(3 downto 0);
RXCHARISK : out std_logic_vector(3 downto 0);
RXCHECKINGCRC : out std_ulogic;
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXCRCERR : out std_ulogic;
RXDATA : out std_logic_vector(31 downto 0);
RXDISPERR : out std_logic_vector(3 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(3 downto 0);
RXREALIGN : out std_ulogic;
RXRECCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(3 downto 0);
TXBUFERR : out std_ulogic;
TXKERR : out std_logic_vector(3 downto 0);
TXN : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(3 downto 0);
BREFCLK : in std_ulogic := 'X';
BREFCLK2 : in std_ulogic := 'X';
CHBOND1 : in std_logic_vector(3 downto 0);
CONFIGENABLE : in std_ulogic;
CONFIGIN : in std_ulogic;
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(3 downto 0);
TXCHARDISPMODE : in std_logic_vector(3 downto 0);
TXCHARDISPVAL : in std_logic_vector(3 downto 0);
TXCHARISK : in std_logic_vector(3 downto 0);
TXDATA : in std_logic_vector(31 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESETE : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT_AURORA_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_AURORA_4 should be placed
-- in architecture after the begin keyword

GT_AURORA_4_INSTANCE_NAME : GT_AURORA_4
-- synopsys translate_off
generic map (ALIGN_COMMA_MSB => boolean_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_OFFSET => integer_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_WAIT => integer_value,
CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLK_COR_KEEP_IDLE => boolean_value,
CLK_COR_REPEAT_WAIT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (
CHBONDDONE => user_CHBONDDONE,
CHBONDO => user_CHBONDO,
CONFIGOUT => user_CONFIGOUT,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCCLK => user_RXRECCCLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CHBONDI => user_CHBONDI,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLKK => user_REFCLKK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDSPVAL => user_TXCHARDSPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

Verilog Instantiation Templates

GT_AURORA_1 Verilog Code

GT_AURORA_1 GT_AURORA_1_name(
.CHBONDDONE (user_CHBONDDONE),
.CHBONDO (user_CHBONDO),
.CONFIGOUT (user_CONFIGOUT),
.RXBUFSTATUS (user_RXBUFSTATUS),
.RXCHARISCOMMA (user_RXCHARISCOMMA),
.RXCHARISK (user_RXCHARISK),
.RXCHECKINGCRC (user_RXCHECKINGCRC),
.RXCLKCORCNT (user_RXCLKCORCNT),
.RXCOMMADET (user_RXCOMMADET),
.RXCRCCERR (user_RXCRCCERR),
.RXDATA (user_RXDATA),
.RXDISPERR (user_RXDISPERR),
.RXLOSSOFSYNC (user_RXLOSSOFSYNC),
.RXNOTINTABLE (user_RXNOTINTABLE),
.RXREALIGN (user_RXREALIGN),
.RXRECCCLK (user_RXRECCCLK),
.RXRUNDISP (user_RXRUNDISP),
.TXBUFERR (user_TXBUFERR),
.TXKERR (user_TXKERR),
.TXN (user_TXN),
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLK (user_BREFCLK),
.BREFCLK2 (user_BREFCLK2),
.CHBONDI (user_CHBONDI),
.CONFIGENABLE (user_CONFIGENABLE),
.CONFIGIN (user_CONFIGIN),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASSBB1OB (user_TXBYPASSBB1OB),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00111110111;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 1;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b11111111111;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000101;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 1;
defparam user_instance_name.RX_DECODER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 1;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

**GT_AURORA_2 Verilog Code**

```
GT_AURORA_2 GT_AURORA_2_name(
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONO (user_CHBONO),
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKCORCNT (user_RXCLKCORCNT),
```
.RXCOMMADET (user_RXCOMMADET),
.RXCRCERR (user_RXCRCERR),
.RXDATA (user_RXDATA),
.RXDISPERR (user_RXDISPERR),
.RXLOSSOFSYNC (user_RXLOSSOFSYNC),
.RXNOTINTABLE (user_RXNOTINTABLE),
.RXREALIGN (user_RXREALIGN),
.RXRECCCLK (user_RXRECCCLK),
.RXRUNDISP (user_RXRUNDISP),
.TXBUFERR (user_TXBUFERR),
.TXKERR (user_TXKERR),
.TXN (user_TXN),
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLK (user_BREFCLK),
.BREFCLK2 (user_BREFCLK2),
.CHBONDI (user_CHBONDI),
.CONFIGENABLE (user_CONFIGENABLE),
.CONFIGIN (user_CONFIGIN),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;

www.xilinx.com
1-800-255-7778
ISE 6.1i

Libraries Guide
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b0001111110111;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b0001111110111;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b0000000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b0000000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b0000000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b0000000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b0000000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b0000000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b11111111111;
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000101;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111010;
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 2;
defparam user_instance_name.RX_DECIMAL_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 2;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

GT_AURORA_4 Verilog Code

```verilog
GT_AURORA_4 GT_AURORA_4_name(
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKKORCNT (user_RXCLKKORCNT),
    .RXCOMMADET (user_RXCOMMADET),
```
.RXCRCERR (user_RXCRCERR),
.RXDATA (user_RXDATA),
.RXDISPERR (user_RXDISPERR),
.RXLOSSOFSYNC (user_RXLOSSOFSYNC),
.RXNOTINTABLE (user_RXNOTINTABLE),
.RXREALIGN (user_RXREALIGN),
.RXRECCCLK (user_RXRECCCLK),
.RXRUNDISP (user_RXRUNDISP),
.TXBUFERR (user_TXBUFERR),
.TXKERR (user_TXKERR),
.TXN (user_TXN),
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLK (user_BREFCLK),
.BREFCLK2 (user_BREFCLK2),
.CHBONDI (user_CHBONDI),
.CONFIGENABLE (user_CONFIGENABLE),
.CONFIGIN (user_CONFIGIN),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00111110111;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00111110111;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00111110111;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00111110111;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 4;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b11111111111;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000101;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 4;
defparam user_instance_name.RX_DECIMATE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 4;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPASIS = 0;

Commonly Used Constraints

None
Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT_CUSTOM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive*</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
<tr>
<td>*Supported for Virtex-II Pro but not for Virtex-II or Virtex-II Pro X.</td>
<td></td>
</tr>
</tbody>
</table>

This gigabit transceiver is fully customizable. You can set attributes for the primitives. You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figure lists the input and output ports for all values of $n$. For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>CHBONO(3:0)</td>
</tr>
<tr>
<td>CHBONDI(3:0)</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>RXCHARISK(3:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(31:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(3:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(3:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISP(mode)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
<td>TXDATA(31:0)</td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td>TXDATA(31:0)</td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TXRESET</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT_CUSTOM VHDL Code

```
-- Component Attribute specification for GT_CUSTOM
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_CUSTOM
    -- synopsys translate_off
    generic (    
        ALIGN_COMMA_MSB           : boolean := FALSE;
        CHAN_BOND_LIMIT           : integer := 16;
        CHAN_BOND_MODE            : string := "OFF";
        CHAN_BOND_OFFSET          : integer := 8;
        CHAN_BOND_ONE_SHOT        : boolean := FALSE;
        CHAN_BOND_SEQ_1_1         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_1_2         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_1_3         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_1_4         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_2_1         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_2_2         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_2_3         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_2_4         : bit_vector := "00000000000";
        CHAN_BOND_SEQ_2_USE       : boolean := FALSE;
        CHAN_BOND_SEQ_LEN         : integer := 1;
        CHAN_BOND_WAIT            : integer := 8;
        CLK_COR_INSERT_IDLE_FLAG  : boolean := FALSE;
        CLK_COR_KEEP_IDLE         : boolean := FALSE;
        CLK_COR_REPEAT_WAIT       : integer := 1;
        CLK_COR_SEQ_1_1           : bit_vector := "00000000000";
        CLK_COR_SEQ_1_2           : bit_vector := "00000000000";
        CLK_COR_SEQ_1_3           : bit_vector := "00000000000";
        CLK_COR_SEQ_1_4           : bit_vector := "00000000000";
        CLK_COR_SEQ_2_1           : bit_vector := "00000000000";
        CLK_COR_SEQ_2_2           : bit_vector := "00000000000";
        CLK_COR_SEQ_2_3           : bit_vector := "00000000000";
        CLK_COR_SEQ_2_4           : bit_vector := "00000000000";
        CLK_COR_SEQ_2_USE         : boolean := FALSE;
        CLK_COR_SEQ_LEN           : integer := 1;
        CLK_CORRECT_USE           : boolean := TRUE;
        COMMA_10B_MASK            : bit_vector := "1111111000";
        CRC_END_OF_PKT            : string := "K29_7";
        CRC_FORMAT                : string := "USER_MODE";
        CRC_START_OF_PKT          : string := "K27_7";
        DEC_MCOMMA_DETECT         : boolean := TRUE;
        DEC_PCOMMA_DETECT         : boolean := TRUE;
        DEC_VALID_COMMA_ONLY      : boolean := TRUE;
        MCOMMA_10B_VALUE          : bit_vector := "11000000000";
        MCOMMA_DETECT             : boolean := TRUE;
        PCOMMA_10B_VALUE          : bit_vector := "00111111000";
        PCOMMA_DETECT             : boolean := TRUE;
        REF_CLK_V_SEL             : integer := 0;
    );
```
RX_BUFFER_USE             : boolean := TRUE;
RX_CRC_USE                : boolean := FALSE;
RX_DATA_WIDTH             : integer := 2;
RX_DECODER_USE            : boolean := TRUE;
RX_LOS_INVALID_INCR       : integer := 1;
RX_LOS_THRESHOLD          : integer := 4;
RX_LOSS_OF_SYNC_FSM       : boolean := TRUE;
SERDES_10B                : boolean := FALSE;
TERMINATION_IMP           : integer := 50;
TX_BUFFER_USE             : boolean := TRUE;
TX_CRC_FORCE_VALUE        : bit_vector := "11010110";
TX_CRC_USE                : boolean := FALSE;
TX_DATA_WIDTH             : integer := 2;
TX_DIFF_CTRL              : integer := 500;
TX_PREEMPHASIS            : integer := 0);

-- synopsys translate_on
port (
  CHBONDDONE               : out std_ulogic;
  CHBONDO                   : out std_logic_vector(3 downto 0);
  CONFIGOUT                 : out std_ulogic;
  RXBUFSTATUS               : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA             : out std_logic_vector(3 downto 0);
  RXCHARISK                 : out std_logic_vector(3 downto 0);
  RXCHECKINGCRC             : out std_ulogic;
  RXCLKCORCNT               : out std_logic_vector(2 downto 0);
  RXCOMMADET                : out std_logic;
  RXCRCERR                  : out std_ulogic;
  RXDATA                    : out std_logic_vector(31 downto 0);
  RXDISPCORCNT              : out std_logic_vector(2 downto 0);
  RXREALIGN                 : out std_ulogic;
  RXRECCLK                  : out std_ulogic;
  RXRUNDISP                 : out std_logic_vector(3 downto 0);
  TXBUFERR                  : out std_ulogic;
  TXKERR                    : out std_logic_vector(3 downto 0);
  TXN                       : out std_ulogic;
  TXP                       : out std_ulogic;
  TXRUNDISP                 : out std_logic_vector(3 downto 0);
  BREFCLK                   : in std_ulogic := 'X';
  BREFCLK2                  : in std_ulogic := 'X';
  CHBONDI                   : in std_logic_vector(3 downto 0);
  CONFIGENABLE              : in std_ulogic;
  CONFIGIN                  : in std_ulogic;
  ENCHANSYNC                : in std_ulogic;
  ENMCOMMAALIGN             : in std_ulogic;
  ENFCOMMAALIGN             : in std_ulogic;
  LOOPBACK                  : in std_logic_vector(1 downto 0);
  POWERDOWN                 : in std_ulogic;
  REFCLK                    : in std_ulogic;
  REFCLK2                   : in std_ulogic;
  REFCLKSEL                 : in std_ulogic;
  RXN                       : in std_ulogic;
  RXP                       : in std_ulogic;
  RXPOLARITY                : in std_ulogic;
  RXRESET                   : in std_ulogic;
  RXUSRCLK                  : in std_ulogic;
)
RXUSRCLK2 : in std_logic;
TXBYPASS&10B : in std_logic_vector(3 downto 0);
TXCHARDISPMODE : in std_logic_vector(3 downto 0);
TXCHARDISPVALL : in std_logic_vector(3 downto 0);
TXCHARK : in std_logic_vector(3 downto 0);
TXDATA : in std_logic_vector(31 downto 0);
TXFORCECRCERR : in std_logic;
TXINHIBIT : in std_logic;
TXPOLARITY : in std_logic;
TXRESET : in std_logic;
TXUSRCLK : in std_logic;
TXUSRCLK2 : in std_logic;)

end component;

-- Component Attribute specification for GT_CUSTOM
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_CUSTOM should be placed
-- in architecture after the begin keyword

GT_CUSTOM_INSTANCE_NAME : GT_CUSTOM
-- synopsys translate_off
generic map ( ALIGN_COMMA_MSB  => boolean_value,
CHAN_BOND_LIMIT  => integer_value,
CHAN_BOND_MODE  => string_value,
CHAN_BOND_OFFSET  => integer_value,
CHAN_BOND_ONE_SHOT  => boolean_value,
CHAN_BOND_SEQ_1_1  => bit_value,
CHAN_BOND_SEQ_1_2  => bit_value,
CHAN_BOND_SEQ_1_3  => bit_value,
CHAN_BOND_SEQ_1_4  => bit_value,
CHAN_BOND_SEQ_1_5  => bit_value,
CHAN_BOND_SEQ_2_1  => bit_value,
CHAN_BOND_SEQ_2_2  => bit_value,
CHAN_BOND_SEQ_2_3  => bit_value,
CHAN_BOND_SEQ_2_4  => bit_value,
CHAN_BOND_SEQ_2_5  => bit_value,
CHAN_BOND_SEQ_2_6  => bit_value,
CHAN_BOND_SEQ_2_7  => bit_value,
CHAN_BOND_SEQ_2_8  => bit_value,
CHAN_BOND_SELED  => boolean_value,
CHAN_BOND_SEQ_LEN  => integer_value,
CHAN_BOND_WAIT  => integer_value,
CLK_COR_INSERT_IDLE_FLAG  => boolean_value,
CLK_COR_KEEP_IDLE  => boolean_value,
CLK_COR_REPEAT_WAIT  => integer_value,
CLK_COR_SEQ_1_1  => bit_value,
CLK_COR_SEQ_1_2  => bit_value,
CLK_COR_SEQ_1_3  => bit_value,
CLK_COR_SEQ_1_4  => bit_value,
CLK_COR_SEQ_2_1  => bit_value,
CLK_COR_SEQ_2_2  => bit_value,
CLK_COR_SEQ_2_3  => bit_value,
CLK_COR_SEQ_2_4  => bit_value,
CLK_COR_SEQ_2_5  => bit_value,
CLK_COR_SEQ_2_6  => bit_value,
CLK_COR_SEQ_2_7  => bit_value,
CLK_COR_SEQ_2_8  => bit_value,
CLK_COR_SEQ_2_LEN  => integer_value,
CLK_CORRECT_USE  => boolean_value,
COMMA_10B_MASK  => bit_value,
-- synopsys translate_on

port map (
  CHBONDDONE => user_CHBONDDONE,
  CHBONDO => user_CHBONDO,
  CONFIGOUT => user_CONFIGOUT,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCHECKINGCRC => user_RXCHECKINGCRC,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXCRCERR => user_RXCRCERR,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOSYNC => user_RXLOSSOSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCCLK => user_RXRECCCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFERR => user_TXBUFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLK => user_BREFCLK,
  BREFCLK2 => user_BREFCLK2,
  CHBONDI => user_CHBONDI,
  CONFIGENABLE => user_CONFIGENABLE,
  CONFIGIN => user_CONFIGIN,
  ENCHANSYNC => user_ENCHANSYNC,
  ENMCOMMAALIGN => user_ENMCOMMAALIGN,
  CRC_END_OF_PKT => string_value,
  CRC_FORMAT => string_value,
  CRC_START_OF_PKT => string_value,
  DEC_MCOMMA_DETECT => boolean_value,
  DEC_PCOMMA_DETECT => boolean_value,
  DEC_VALID_COMMA_ONLY => boolean_value,
  MCOMMA_10B_VALUE => bit_value,
  MCOMMA_DETECT => boolean_value,
  PCOMMA_10B_VALUE => bit_value,
  PCOMMA_DETECT => boolean_value,
  REF_CLK_V_SEL => integer_value,
  RX_BUFFER_USE => boolean_value,
  RX_CRC_USE => boolean_value,
  RX_DATA_WIDTH => integer_value,
  RX_DECODE_USE => boolean_value,
  RX_LOS_INVALID_INCR => integer_value,
  RX_LOS_THRESHOLD => integer_value,
  RX_LOSS_OF_SYNC_FSM => boolean_value,
  SERDES_10B => boolean_value,
  TERMINATION_IMP => integer_value,
  TX_BUFFER_USE => boolean_value,
  TX_CRC_FORCE_VALUE => bit_value,
  TX_CRC_USE => boolean_value,
  TX_DATA_WIDTH => integer_value,
  TX_DIFF_CTRL => integer_value,
  TX_PREEMPHASIS => integer_value)

740
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXXDATA => user_TXXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

Verilog Instantiation Templates

GT_CUSTOM Verilog Code

GT_CUSTOM GT_CUSTOM_name(
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .CONFIGOUT (user_CONFIGOUT),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCHECKINGCRC (user_RXCHECKINGCRC),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXCRCERR (user_RXCRCERR),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSOFSYNC (user_RXLOSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKKSEL (user_REFCLKKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLC_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLC_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLC_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLC_COR_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLC_COR_SEQ_LEN = 1;
defparam user_instance_name.CLC_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111000;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 2;
defparam user_instance_name.RX_DECIMATE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 2;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

Commonly Used Constraints

None
GT_ETHERNET_n

Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT_ETHERNET_n</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X Primitive*</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro but not for Virtex-II or Virtex-II Pro X.

This Ethernet gigabit transceiver supports 1, 2, and 4-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>GT_ETHERNET_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
</tr>
<tr>
<td>BREFCLK2</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
</tr>
<tr>
<td>CONFIGIN</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>POWERTDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLKSEL</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPVAL(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
</tbody>
</table>
### Usage

This design element is instantiated rather than inferred in the design code.
VHDL Instantiation Templates

GT_ETHERNET_1 VHDL Code

-- Component Attribute specification for GT_ETHERNET_1
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_ETHERNET_1
-- synopsys translate_off
generic (  
  ALIGN_COMMA_MSB : boolean := FALSE;
  CHAN_BOND_LIMIT : integer := 1;
  CHAN_BOND_MODE : string := "OFF";
  CHAN_BOND_OFFSET : integer := 0;
  CHAN_BOND_ONE_SHOT : boolean := TRUE;
  CHAN_BOND_SEQ_1_1 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_USE : boolean := FALSE;
  CHAN_BOND_SEQ_LEN : integer := 1;
  CHAN_BOND_WAIT : integer := 7;
  CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
  CLK_COR_KEEP_IDLE : boolean := FALSE;
  CLK_COR_REPEAT_WAIT : integer := 1;
  CLK_COR_SEQ_1_1 : bit_vector := "00110111100";
  CLK_COR_SEQ_1_2 : bit_vector := "00010100000";
  CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
  CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_USE : boolean := FALSE;
  CLK_COR_SEQ_LEN : integer := 2;
  CLK_CORRECT_USE : boolean := TRUE;
  COMMA_10B_MASK : bit_vector := "1111111000";
  CRC_END_OF_PKT : string := "K29_7";
  CRC_FORMAT : string := "ETHERNET";
  CRC_START_OF_PKT : string := "K27_7";
  DEC_MCOMMA_DETECT : boolean := TRUE;
  DEC_PCOMMA_DETECT : boolean := TRUE;
  DEC_VALID_COMMA_ONLY : boolean := TRUE;
  MCOMMA_10B_VALUE : bit_vector := "1111111000";
  MCOMMA_DETECT : boolean := TRUE;
  PCOMMA_10B_VALUE : bit_vector := "0011111000";
  PCOMMA_DETECT : boolean := TRUE;
  REF_CLK_V_SEL : integer := 0;
  RX_BUFFER_USE : boolean := TRUE;
  RX_CRC_USE : boolean := FALSE;
  RX_DATA_WIDTH : integer := 1;
  RX_DECODE_USE : boolean := TRUE;  
)
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 1;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port :

CONFIGOUT : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(0 downto 0);
RXCHARISK : out std_logic_vector(0 downto 0);
RXCHECKINGCRC : out std_logic;
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXCRCERR : out std_ulogic;
RXDATA : out std_logic_vector(7 downto 0);
RXDISPERR : out std_logic_vector(0 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(0 downto 0);
RXREALIGN : out std_ulogic;
RXRECLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(0 downto 0);
TXBUFERR : out std_ulogic;
TXKERR : out std_logic_vector(0 downto 0);
TXN : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(0 downto 0);
BREFCLK : in std_logic := 'X';
BREFCLK2 : in std_logic := 'X';
CONFIGENABLE : in std_ulogic;
CONFIGIN : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(0 downto 0);
TXCHARDISPMODE : in std_logic_vector(0 downto 0);
TXCHARDISPVAL : in std_logic_vector(0 downto 0);
TXCHARISK : in std_logic_vector(0 downto 0);
TxDATA : in std_logic_vector(7 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
end component;

-- Component Attribute specification for GT_ETHERNET_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_ETHERNET_1 should be placed
-- in architecture after the begin keyword

GT_ETHERNET_1_INSTANCE_NAME : GT_ETHERNET_1

-- synopsys translate_off
generic map (ALIGN_COMMA_MSB           => boolean_value,
            CHAN_BOND_LIMIT           => integer_value,
            CHAN_BOND_MODE            => string_value,
            CHAN_BOND_OFFSET          => integer_value,
            CHAN_BOND_ONE_SHOT        => boolean_value,
            CHAN_BOND_SEQ_1_1         => bit_value,
            CHAN_BOND_SEQ_1_2         => bit_value,
            CHAN_BOND_SEQ_1_3         => bit_value,
            CHAN_BOND_SEQ_1_4         => bit_value,
            CHAN_BOND_SEQ_2_1         => bit_value,
            CHAN_BOND_SEQ_2_2         => bit_value,
            CHAN_BOND_SEQ_2_3         => bit_value,
            CHAN_BOND_SEQ_2_4         => bit_value,
            CHAN_BOND_SEQ_2_USE       => boolean_value,
            CHAN_BOND_SEQ_LEN         => integer_value,
            CHAN_BOND_WAIT            => integer_value,
            CLK_COR_INSERT_IDLE_FLAG  => boolean_value,
            CLK_COR_KEEP_IDLE         => boolean_value,
            CLK_COR_REPEAT_WAIT       => integer_value,
            CLK_COR_SEQ_1_1           => bit_value,
            CLK_COR_SEQ_1_2           => bit_value,
            CLK_COR_SEQ_1_3           => bit_value,
            CLK_COR_SEQ_1_4           => bit_value,
            CLK_COR_SEQ_2_1           => bit_value,
            CLK_COR_SEQ_2_2           => bit_value,
            CLK_COR_SEQ_2_3           => bit_value,
            CLK_COR_SEQ_2_4           => bit_value,
            CLK_COR_SEQ_2_USE         => boolean_value,
            CLK_COR_SEQ_LEN           => integer_value,
            CLK_CORRECT_USE           => boolean_value,
            COMMA_10B_MASK            => bit_value,
            CRC_END_OF_PKT            => string_value,
            CRC_FORMAT                => string_value,
            CRC_START_OF_PKT          => string_value,
            DEC_MCOMMA_DETECT         => boolean_value,
            DEC_PCOMMA_DETECT         => boolean_value,
            DEC_VALID_COMMA_ONLY      => boolean_value,
            MCOMMA_10B_VALUE          => bit_value,
            MCOMMA_DETECT             => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (CONFIGOUT => user_CONFIGOUT,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RCLKCORCNT => user_RCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => userPOWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TxUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT_ETHERNET_2 VHDL Code

-- Component Attribute specification for GT_ETHERNET_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_ETHERNET_2
-- synopsys translate_off
generic (ALIGN_COMMA_MSB : boolean := FALSE;
CHAN_BOND_LIMIT : integer := 1;
CHAN_BOND_MODE : string := "OFF";
CHAN_BOND_OFFSET : integer := 0;
CHAN_BOND_ONE_SHOT : boolean := TRUE;
CHAN_BOND_SEQ_1_1 : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_USE : boolean := FALSE;
CHAN_BOND_SEQ_LEN : integer := 1;
CHAN_BOND_WAIT : integer := 7;
CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
CLK_COR_KEEP_IDLE : boolean := FALSE;
CLK_COR_REPEAT_WAIT : integer := 1;
CLK_COR_SEQ_1_1 : bit_vector := "00110111100";
CLK_COR_SEQ_1_2 : bit_vector := "00001010000";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 2;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "1111111000";
CRC_END_OF_PKT : string := "K29_7";
CRC_FORMAT : string := "ETHERNET";
CRC_START_OF_PKT : string := "K27_7";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE          : bit_vector := "1100000000";
MCOMMA_DETECT             : boolean := TRUE;
PCOMMA_10B_VALUE          : bit_vector := "0011111000";
PCOMMA_DETECT             : boolean := TRUE;
REF_CLK_V_SEL             : integer := 0;
RX_BUFFER_USE             : boolean := TRUE;
RX_CRC_USE                : boolean := FALSE;
RX_DATA_WIDTH             : integer := 2;
RX_DECODE_USE             : boolean := TRUE;
RX_LOS_INVALID_INCR       : integer := 1;
RX_LOS_THRESHOLD          : integer := 4;
RX_LOSS_OF_SYNC_FSM       : boolean := TRUE;
SERDES_10B                : boolean := FALSE;
TERMINATION_IMP           : integer := 50;
TX_BUFFER_USE             : boolean := TRUE;
TX_CRC_FORCE_VALUE        : bit_vector := "11010110";
TX_CRC_USE                : boolean := FALSE;
TX_DATA_WIDTH             : integer := 2;
TX_DIFF_CTRL              : integer := 500;
RX_PREEMPHASIS            : integer := 0);

port (  
  CONFIGOUT               : out std_ulogic;
  RXBUFSTATUS          : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA       : out std_logic_vector(1 downto 0);
  RXCHARISK           : out std_logic_vector(1 downto 0);
  RXCHECKINGCRC       : out std_logic;
  RXCLKCORCNT        : out std_logic_vector(2 downto 0);
  RXCOMMADET          : out std_logic;
  RXCRCERR            : out std_ulogic;
  RXDATA              : out std_logic_vector(15 downto 0);
  RXDISPERR           : out std_logic_vector(1 downto 0);
  RXLOSSOFSYNC       : out std_logic_vector(1 downto 0);
  RXNOTINTABLE       : out std_logic_vector(1 downto 0);
  RXREALIGN            : out std_logic;
  RXRECCLOCK        : out std_logic;
  RXRUNDISP         : out std_logic_vector(1 downto 0);
  TXBUFERR             : out std_ulogic;
  TXKERR              : out std_logic_vector(1 downto 0);
  TXN                 : out std_ulogic;
  TXP                 : out std_ulogic;
  TXRUNDISP         : out std_logic_vector(1 downto 0);
  BREFCLK             : in std_ulogic := 'X';
  BREFCLK2            : in std_ulogic := 'X';
  CONFIGENABLE        : in std_ulogic;
  CONFIGIN             : in std_ulogic;
  ENMCOMMAALIGN       : in std_logic;
  ENPCOMMAALIGN       : in std_logic;
  LOOPBACK            : in std_logic_vector(1 downto 0);
  POWERDOWN           : in std_ulogic;
  REFCLK              : in std_ulogic;
  REFCLK2             : in std_ulogic;
  REFCLKSEL           : in std_ulogic;
  RXN                 : in std_ulogic;
  RXP                 : in std_ulogic;
  RXPOLARITY          : in std_logic;
  RXRESET             : in std_logic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(1 downto 0);
TXCHARDISPMODE : in std_logic_vector(1 downto 0);
TXCHARDISPVAL : in std_logic_vector(1 downto 0);
TXCHARISK : in std_logic_vector(1 downto 0);
TXDATA : in std_logic_vector(15 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT_ETHERNET_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_ETHERNET_2 should be placed
-- in architecture after the begin keyword

GT_ETHERNET_2_INSTANCE_NAME : GT_ETHERNET_2

-- synopsys translate_off
generic map (ALIGN_COMMA_MSB => boolean_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_OFFSET => integer_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_WAIT => integer_value,
CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLK_COR_KEEP_IDLE => boolean_value,
CLK_COR_REPEAT_WAIT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
-- synopsys translate_on
port map (  
  CONFIGOUT => user_CONFIGOUT,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCHECKINGCRC => user_RXCHECKINGCRC,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXCRCERR => user_RXCRCERR,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCCLK => user_RXRECCCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFERR => user_TXBUFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLK => user_BREFCLK,
  BREFCLK2 => user_BREFCLK2,
  CONFIGENABLE => user_CONFIGENABLE,
  CONFIGIN => user_CONFIGIN,
  ENMCOMMAALIGN => user_ENMCOMMAALIGN,
  ENPCOMMAALIGN => user_ENPCOMMAALIGN,
  LOOPBACK => user_LOOPBACK,
  POWERDOWN => user_POWERDOWN,
GT_ETHERNET_n

REFCLK         => user_REFCLK,
REFCLK2        => user_REFCLK2,
REFCLKSEL      => user_REFCLKSEL,
RXN            => user_RXN,
RXP            => user_RXP,
RXPOLARITY     => user_RXPOLARITY,
RXRESET        => user_RXRESET,
RXUSRCLK       => user_RXUSRCLK,
RXUSRCLK2      => user_RXUSRCLK2,
TXBYPASS8B10B  => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL  => user_TXCHARDISPVAL,
TXCHARISK      => user_TXCHARISK,
TXDATA         => user_TXDATA,
TXFORCECRCERR  => user_TXFORCECRCERR,
TXINHIBIT      => user_TXINHIBIT,
TXPOLARITY     => user_TXPOLARITY,
TXRESET        => user_TXRESET,
TXUSRCLK       => user_TXUSRCLK,
TXUSRCLK2      => user_TXUSRCLK2);

GT_ETHERNET_4 VHDL Code

-- Component Attribute specification for GT_ETHERNET_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_ETHERNET_4
-- synopsys translate_off
generic (ALIGN_COMMA_MSB           : boolean := FALSE;
CHAN_BOND_LIMIT           : integer := 1;
CHAN_BOND_MODE            : string := "OFF";
CHAN_BOND_OFFSET          : integer := 0;
CHAN_BOND_ONE_SHOT        : boolean := TRUE;
CHAN_BOND_SEQ_1_1         : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_2         : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_3         : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_4         : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_1         : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_2         : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_3         : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_4         : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_USE       : boolean := FALSE;
CHAN_BOND_SEQ_LEN         : integer := 1;
CHAN_BOND_WAIT            : integer := 1;
CLK_COR_INSERT_IDLE_FLAG  : boolean := FALSE;
CLK_COR_KEEP_IDLE         : boolean := FALSE;
CLK_COR_REPEAT_WAIT       : integer := 1;
CLK_COR_SEQ_1_1           : bit_vector := "00110111100";
CLK_COR_SEQ_1_2           : bit_vector := "00010100000";
CLK_COR_SEQ_1_3           : bit_vector := "00000000000";
CLK_COR_SEQ_1_4           : bit_vector := "00000000000";
CLK_COR_SEQ_2_1           : bit_vector := "00000000000";
CLK_COR_SEQ_2_2           : bit_vector := "00000000000";
CLK_COR_SEQ_2_3           : bit_vector := "00000000000";
CLK_COR_SEQ_2_4           : bit_vector := "00000000000";
CLK_COR_SEQ_2_USE         : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 2;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "1111111000";
CRC_END_OF_PKT : string := "K29_7";
CRC_FORMAT : string := "ETHERNET";
CRC_START_OF_PKT : string := "K27_7";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "1100000000";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "0011111000";
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 4;
RX_DECODE_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 4;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0;

-- synopsys translate_on

-- synopsys translate_off

port ( CONFIGOUT : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISKOMMA : out std_logic_vector(3 downto 0);
RXCHARISK : out std_logic_vector(3 downto 0);
RXCHECKINGCRC : out std_logic;
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXCRCERR : out std_ulogic;
RXDATA : out std_logic_vector(31 downto 0);
RXDISPERR : out std_logic_vector(3 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(3 downto 0);
RXREALIGN : out std_ulogic;
RXRECCCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(3 downto 0);
TXBUFERR : out std_ulogic;
TXKERR : out std_logic_vector(3 downto 0);
TXN : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(3 downto 0);
BREFCLK : in std_ulogic := 'X';
BREFCLK2 : in std_ulogic := 'X';
CONFIGENABLE : in std_ulogic;
CONFIGIN : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXN : in std_ulogic;
RXF : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(3 downto 0);
TXCHARDISPMODE : in std_logic_vector(3 downto 0);
TXCHARDISPVAL : in std_logic_vector(3 downto 0);
TXCHARISK : in std_logic_vector(3 downto 0);
TXDATA : in std_logic_vector(31 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT_ETHERNET_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_ETHERNET_4 should be placed
-- in architecture after the begin keyword

GT_ETHERNET_4_INSTANCE_NAME : GT_ETHERNET_4

-- synopsys translate_off

generic map (ALIGN_COMMA_MSB => boolean_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_OFFSET => integer_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_WAIT => integer_value,
CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLK_COR_KEEP_IDLE => boolean_value,
CLK_COR_REPEAT_WAIT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DIFF_CTRL => integer_value,
RX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (}
CONFIGOUT => user_CONFIGOUT,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECClk => user_RXRECClk,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CONFIGENABLE => userCONFIGENABLE,
CONFIGIN => userCONFIGIN,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPFCOMMAALIGN => user_ENPFCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

Verilog Instantiation Templates

GT_ETHERNET_1 Verilog Code

GT_ETHERNET_1 GT_ETHERNET_1_name(
  .CONFIGOUT (user_CONFIGOUT),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCHECKINGCRC (user_RXCHECKINGCRC),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXCRCERR (user_RXCRCERR),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOSYNTAX (user_RXLOSSOSYNTAX),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
.BREFCLK2 (user_BREFCLK2),
.CONFIGENABLE (user_CONFIGENABLE),
.CONFIGIN (user_CONFIGIN),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 1;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 0;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 7;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b0011011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00011110000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111000;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "ETHERNET";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 1;
defparam user_instance_name.RX_DECIDE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 1;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

GT_ETHERNET_2 Verilog Code

GT_ETHERNET_2 GT_ETHERNET_2_name(
  .CONFIGOUT (user_CONFIGOUT),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCHECKINGCRC (user_RXCHECKINGCRC),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXCRCERR (user_RXCRCERR),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXERR (user_TXERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (userPOWERDOWN),
.REFCLK (userREFCLK),
.REFCLK2 (userREFCLK2),
.REFCLKSEL (userREFCLKSEL),
.RXN (userRXN),
.RXP (userRXP),
.RXPOLARITY (userRXPOLARITY),
.RXRESET (userRXRESET),
.RXUSRCLK (userRXUSRCLK),
.RXUSRCLK2 (userRXUSRCLK2),
.TXBYPAS8B10B (userTXBYPAS8B10B),
.TXCHARDISPMODE (userTXCHARDISPMODE),
.TXCHARDISPVAL (userTXCHARDISPVAL),
.TXCHARISK (userTXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 1;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 0;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 7;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00001010000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111100;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "ETHERNET";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEQ_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0011111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 2;
defparam user_instance_name.RX_DECODE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 2;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

GT_Ethernet_4 Verilog Code

GT_Ethernet_4 GT_Ethernet_4_name(
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXCRCERR (user_RXCRCERR),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSOFSYNC (user_RXLOSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLK (user_BREFCLK),
    .BREFCLK2 (user_BREFCLK2),
    .CONFIGENABLE (user_CONFIGENABLE),
    .CONFIGIN (user_CONFIGIN),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .POWERDOWN (user_POWERDOWN),
    .REFCLK (user_REFCLK),
    .REFCLK2 (user_REFCLK2),
    .REFCLKSEL (user_REFCLKSEL),
)
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODEL (user_TXCHARDISPMODEL),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

```vhdl
defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 1;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 0;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 7;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b001110111100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00001010000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111000;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "ETHERNET";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1111111000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111100;
```
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 4;
defparam user_instance_name.RX_DECODE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 4;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

Commonly Used Constraints

None
GT_FIBRE_CHAN_n

Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>GT_FIBRERE_CHAN_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive*</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro but not for Virtex-II or Virtex-II Pro X.

This Fibre Channel gigabit transceiver supports 1, 2, and 4-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the RocketIO Transceiver User Guide.
### GT_FIBRE_CHAN_1

<table>
<thead>
<tr>
<th>Signal</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>RXBUFSIZE(1:0)</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXCHARISCOMMA(0:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISK(0:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(7:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(0:0)</td>
</tr>
<tr>
<td>REFLKSEL</td>
<td>RXLOSSOSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(0:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(0:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(0:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(0:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(0:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPMODE(0:0)</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXCHARISK(0:0)</td>
<td></td>
</tr>
<tr>
<td>TDATA(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TX_RESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
### GT_FIBRE_CHAN_2

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISK(1:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(15:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(1:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(1:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(1:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(1:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPMODEVAL(1:0)</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
<tr>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>BREFCLK</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISK(3:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(31:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(3:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(3:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPMODEVAL(3:0)</td>
<td>TXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

**GT_FIBRE_CHAN_1 VHDL Code**

```vhdl
-- Component Attribute specification for GT_FIBRE_CHAN_1
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT_FIBRE_CHAN_1
-- synopsys translate_off
BEGIN
  -- Component Attribute specification for GT_FIBRE_CHAN_1
  -- should be placed after architecture declaration but
  -- before the begin keyword
  GENERIC
    ALIGN_COMMA_MSB : boolean := FALSE;
    CHAN_BOND_LIMIT : integer := 1;
    CHAN_BOND_MODE  : string  := "OFF";
    CHAN_BOND_OFFSET : integer := 0;
    CHAN_BOND_ONE_SHOT : boolean := TRUE;
    CHAN_BOND_SEQ_1_1 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_2 : bit_vector := "10000000000";
    CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_USE : boolean := FALSE;
    CHAN_BOND_SEQ_LEN : integer := 1;
    CHAN_BOND_WAIT : integer := 7;
    CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
    CLK_COR_KEEP_IDLE : boolean := FALSE;
    CLK_COR_REPEAT_WAIT : integer := 2;
    CLK_COR_SEQ_1_1 : bit_vector := "00110111100";
    CLK_COR_SEQ_1_2 : bit_vector := "00010010101";
    CLK_COR_SEQ_1_3 : bit_vector := "00010110101";
    CLK_COR_SEQ_1_4 : bit_vector := "00010110101";
    CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
    CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
    CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
    CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
    CLK_COR_SEQ_2_USE : boolean := FALSE;
    CLK_COR_SEQ_LEN : integer := 4;
    CLK_CORRECT_USE : boolean := TRUE;
    COMMA_10B_MASK : bit_vector := "1111111100";
    CRC_END_OF_PKT : string := "K29_7";
    CRC_FORMAT : string := "FIBRE_CHAN";
    CRC_START_OF_PKT : string := "K27_7";
    DEC_MCOMMA_DETECT : boolean := TRUE;
    DEC_PCOMMA_DETECT : boolean := TRUE;
    DEC_VALID_COMMA_ONLY : boolean := TRUE;
    MCOMMA_10B_VALUE : bit_vector := "11000000000";
    MCOMMA_DETECT : boolean := TRUE;
    PCOMMA_10B_VALUE : bit_vector := "00111111000";
    PCOMMA_DETECT : boolean := TRUE;
    REF_CLK_V_SEL : integer := 0;
END COMPONENT;
```

RX_BUFFER_USE       : boolean := TRUE;
RX_CRC_USE          : boolean := FALSE;
RX_DATA_WIDTH       : integer := 1;
RX_DECODE_USE       : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD    : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B          : boolean := FALSE;
TERMINATION_IMP     : integer := 50;
TX_BUFFER_USE       : boolean := TRUE;
TX_CRC_FORCE_VALUE  : bit_vector := "11010110";
TX_CRC_USE          : boolean := FALSE;
TX_DATA_WIDTH       : integer := 1;
TX_DIFF_CTRL        : integer := 500;
TX_PREEMPHASIS      : integer := 0);

-- synopsys translate_on
port (           
CONFIGOUT                 : out std_ulogic;
RXBUFSTATUS               : out std_logic_vector(1 downto 0);
RXCHARISCOMMA             : out std_logic_vector(0 downto 0);
RXCHARISK                : out std_logic_vector(0 downto 0);
RXCHECKINGCRC             : out std_logic;
RXCLKCORCNT              : out std_logic_vector(2 downto 0);
RXCOMMADET               : out std_ulogic;
RXCRCERR                 : out std_ulogic;
RXDATA                    : out std_logic_vector(7 downto 0);
RXDISPERR                : out std_logic_vector(0 downto 0);
RXLOSSOFSYNC             : out std_logic_vector(1 downto 0);
RXNOTINTABLE             : out std_logic_vector(0 downto 0);
RXREALIGN                : out std_ulogic;
RXRECLK                  : out std_logic;
RXRUNDISP                : out std_logic_vector(0 downto 0);
TXBUFERR                 : out std_logic;
TXKERR                   : out std_logic_vector(0 downto 0);
TXN                      : out std_logic;
TXP                      : out std_logic;
TXRUNDISP                : out std_logic_vector(0 downto 0);
BREFCLK                  : in std_logic := 'X';
BREFCLK2                 : in std_logic := 'X';
CONFIGENABLE             : in std_logic;
CONFIGIN                 : in std_logic;
ENKCOMMAALIGN            : in std_logic;
ENPCOMMAALIGN            : in std_logic;
LOOPBACK                 : in std_logic_vector(1 downto 0);
POWERDWN                 : in std_logic;
REFCLK                   : in std_logic;
REFCLK2                  : in std_logic;
REFCLKSEL                : in std_logic;
RXN                      : in std_logic;
RXP                      : in std_logic;
RXPOLARITY               : in std_logic;
RXRESET                  : in std_logic;
RXUSRCLK                 : in std_logic;
RXUSRCLK2                : in std_logic;
TXBYPASS8B10B            : in std_logic_vector(0 downto 0);
TXCHARDISPMODE           : in std_logic_vector(0 downto 0);
TXCHARDISPVAL            : in std_logic_vector(0 downto 0);
TXCHARISK : in std_logic_vector(0 downto 0);
TXDATA     : in std_logic_vector(7 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT  : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESET    : in std_ulogic;
TXUSRCLK   : in std_ulogic;
TXUSRCLK2  : in std_ulogic);

end component;

-- Component Attribute specification for GT_FIBRE_CHAN_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_FIBRE_CHAN_1 should be placed
-- in architecture after the begin keyword

GT_FIBRE_CHAN_1_INSTANCE_NAME : GT_FIBRE_CHAN_1
-- synopsys translate_off
generic map (ALIGN_COMMA_MSB           => boolean_value,
CHAN_BOND_LIMIT           => integer_value,
CHAN_BOND_MODE            => string_value,
CHAN_BOND_OFFSET          => integer_value,
CHAN_BOND_ONE_SHOT        => boolean_value,
CHAN_BOND_SEQ_1_1         => bit_value,
CHAN_BOND_SEQ_1_2         => bit_value,
CHAN_BOND_SEQ_1_3         => bit_value,
CHAN_BOND_SEQ_1_4         => bit_value,
CHAN_BOND_SEQ_2_1         => bit_value,
CHAN_BOND_SEQ_2_2         => bit_value,
CHAN_BOND_SEQ_2_3         => bit_value,
CHAN_BOND_SEQ_2_4         => bit_value,
CHAN_BOND_SEQ_2_USE       => boolean_value,
CHAN_BOND_SEQ_LEN         => integer_value,
CHAN_BOND_WAIT            => integer_value,
CLK_COR_INSERT_IDLE_FLAG  => boolean_value,
CLK_COR_KEEP_IDLE         => boolean_value,
CLK_COR_REPEAT_WAIT       => integer_value,
CLK_COR_SEQ_1_1           => bit_value,
CLK_COR_SEQ_1_2           => bit_value,
CLK_COR_SEQ_1_3           => bit_value,
CLK_COR_SEQ_1_4           => bit_value,
CLK_COR_SEQ_2_1           => bit_value,
CLK_COR_SEQ_2_2           => bit_value,
CLK_COR_SEQ_2_3           => bit_value,
CLK_COR_SEQ_2_4           => bit_value,
CLK_COR_SEQ_2_USE         => boolean_value,
CLK_COR_SEQ_LEN           => integer_value,
CLK_CORRECT_USE           => boolean_value,
COMMA_10B_MASK            => bit_value,
CRC_END_OF_PKT            => string_value,
CRC_FORMAT               => string_value,
CRC_START_OF_PKT          => string_value,
DEC_MCOMMA_DETECT         => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (
    CONFIGOUT => user_CONFIGOUT,
    RXBUFFSTATUS => user_RXBUFFSTATUS,
    RXCHARISCOMMA => user_RXCHARISCOMMA,
    RXCHARISK => user_RXCHARISK,
    RXCHECKINGCRC => user_RXCHECKINGCRC,
    RXCLKCORCNT => user_RXCLKCORCNT,
    RXCOMMADET => user_RXCOMMADET,
    RXCRCERR => user_RXCRCERR,
    RXDATA => user_RXDATA,
    RXDISPERR => user_RXDISPERR,
    RXLOSSOFSYNC => user_RXLOSSOFSYNC,
    RXNOTINTABLE => user_RXNOTINTABLE,
    RXREALIGN => user_RXREALIGN,
    RXRECLK => user_RXRECLK,
    RXRUNDISP => user_RXRUNDISP,
    TXBUFFERR => user_TXBUFFERR,
    TXKERR => user_TXKERR,
    TXN => user_TXN,
    TXP => user_TXP,
    TXRUNDISP => user_TXRUNDISP,
    BREFCLK => user_BREFCLK,
    BREFCLK2 => user_BREFCLK2,
    CONFIGENABLE => user_CONFIGENABLE,
    CONFIGIN => user_CONFIGIN,
    ENMCOMMAALIGN => user_ENMCOMMAALIGN,
    ENPCOMMAALIGN => user_ENPCOMMAALIGN,
    LOOPBACK => user_LOOPBACK,
    POWERDOWN => user_POWERDOWN,
    REFCLK => user_REFCLK,
    REFCLK2 => user_REFCLK2,
    REFCLKSEL => user_REFCLKSEL,
    RXN => user_RXN,
    RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT_FIBRE_CHAN_2 VHDL Code

-- Component Attribute specification for GT_FIBRE_CHAN_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_FIBRE_CHAN_2
-- synopsys translate_off
generic (  
   ALIGN_COMMA_MSB : boolean := FALSE;
   CHAN_BOND_LIMIT : integer := 1;
   CHAN_BOND_MODE : string := "OFF";
   CHAN_BOND_OFFSET : integer := 0;
   CHAN_BOND_ONE_SHOT : boolean := TRUE;
   CHAN_BOND_SEQ_1_1 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_2_USE : boolean := FALSE;
   CHAN_BOND_SEQ_LEN : integer := 1;
   CHAN_BOND_WAIT : integer := 7;
   CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
   CLK_COR_KEEP_IDLE : boolean := FALSE;
   CLK_COR_REPEAT_WAIT : integer := 2;
   CLK_COR_SEQ_1_1 : bit_vector := "00110111100";
   CLK_COR_SEQ_1_2 : bit_vector := "00010010101";
   CLK_COR_SEQ_1_3 : bit_vector := "00010110101";
   CLK_COR_SEQ_1_4 : bit_vector := "00010110101";
   CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_USE : boolean := FALSE;
   CLK_COR_SEQ_LEN : integer := 4;
   CLK_CORRECT_USE : boolean := TRUE;
   COMMA_10B_MASK : bit_vector := "1111111100";
   CRC_END_OF_PKT : string := "K29_7";
   CRC_FORMAT : string := "FIBRE_CHAN";  
);
--- synopsys translate_on

port (  
  CONFIGOUT : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(1 downto 0);
  RXCHARISK : out std_logic_vector(1 downto 0);
  RXCHECKINGCRC : out std_ulogic;
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_ulogic;
  RXCRCERR : out std_ulogic;
  RXDATA : out std_logic_vector(15 downto 0);
  RXDISPERR : out std_logic_vector(1 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(1 downto 0);
  RXREALIGN : out std_ulogic;
  RXRECCCLK : out std_ulogic;
  RXRUNDISP : out std_logic_vector(1 downto 0);
  TXBUFERR : out std_ulogic;
  TXKERR : out std_logic_vector(1 downto 0);
  TXN : out std_ulogic;
  TXP : out std_ulogic;
  TXRUNDISP : out std_logic_vector(1 downto 0);
  BREFCLK : in std_ulogic := 'X';
  BREFCLK2 : in std_ulogic := 'X';
  CONFIGENABLE : in std_ulogic;
  CONFIGIN : in std_ulogic;
  ENMCOMMAALIGN : in std_ulogic;
  ENPCOMMAALIGN : in std_ulogic;
  LOOPBACK : in std_logic_vector(1 downto 0);
  POWERDOWN : in std_ulogic;
  REFCLK : in std_ulogic;
  REFCLK2 : in std_ulogic;
  REFCLKSEL : in std_ulogic;
);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(1 downto 0);
TXCHARDISPMODE : in std_logic_vector(1 downto 0);
TXCHARDISPVAL : in std_logic_vector(1 downto 0);
TXCHARISK : in std_logic_vector(1 downto 0);
TXDATA : in std_logic_vector(15 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT_FIBRE_CHAN_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_FIBRE_CHAN_2 should be placed
-- in architecture after the begin keyword

GT_FIBRE_CHAN_2_INSTANCE_NAME : GT_FIBRE_CHAN_2
-- synopsys translate_off
generic map ( 
  ALIGN_COMMA_MSB => boolean_value,
  CHAN_BOND_LIMIT => integer_value,
  CHAN_BOND_MODE => string_value,
  CHAN_BOND_OFFSET => integer_value,
  CHAN_BOND_ONE_SHOT => boolean_value,
  CHAN_BOND_SEQ_1_1 => bit_value,
  CHAN_BOND_SEQ_1_2 => bit_value,
  CHAN_BOND_SEQ_1_3 => bit_value,
  CHAN_BOND_SEQ_1_4 => bit_value,
  CHAN_BOND_SEQ_2_1 => bit_value,
  CHAN_BOND_SEQ_2_2 => bit_value,
  CHAN_BOND_SEQ_2_3 => bit_value,
  CHAN_BOND_SEQ_2_4 => bit_value,
  CHAN_BOND_SEQ_2_USE => boolean_value,
  CHAN_BOND_SEQ_LEN => integer_value,
  CHAN_BOND_WAIT => integer_value,
  CLK_COR_INSERT_IDLE_FLAG => boolean_value,
  CLK_COR_KEEP_IDLE => boolean_value,
  CLK_COR_REPEAT_WAIT => integer_value,
  CLK_COR_SEQ_1_1 => bit_value,
  CLK_COR_SEQ_1_2 => bit_value,
  CLK_COR_SEQ_1_3 => bit_value,
  CLK_COR_SEQ_1_4 => bit_value,
  CLK_COR_SEQ_2_1 => bit_value,
  CLK_COR_SEQ_2_2 => bit_value,
  CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (}
CONFIGOUT => user_CONFIGOUT,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCLOCK => user_RXRECCLOCK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
GT_FIBRE_CHAN_4 VHDL Code

-- Component Attribute specification for GT_FIBRE_CHAN_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_FIBRE_CHAN_4
  -- synopsys translate_off
  generic (ALIGN_COMMA_MSB : boolean := FALSE;
            CHAN_BOND_LIMIT : integer := 1;
            CHAN_BOND_MODE : string := "OFF";
            CHAN_BOND_OFFSET : integer := 0;
            CHAN_BOND_ONE_SHOT : boolean := TRUE;
            CHAN_BOND_SEQ_1_1 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
            CHAN_BOND_SEQ_2_USE : boolean := FALSE;
            CHAN_BOND_SEQ_LEN : integer := 1;
            CHAN_BOND_WAIT : integer := 7;
            CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
            CLK_COR_KEEP_IDLE : boolean := FALSE;
            CLK_COR_REPEAT_WAIT : integer := 2;
            CLK_COR_SEQ_1_1 : bit_vector := "00110111100";
            CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
            CLK_COR_SEQ_1_3 : bit_vector := "00010110101";
            CLK_COR_SEQ_1_4 : bit_vector := "00010110101";
            CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
            CLK_COR_SEQ_2_2 : bit_vector := "00110111100";
            CLK_COR_SEQ_2_3 : bit_vector := "000010010101";
            CLK_COR_SEQ_2_4 : bit_vector := "00010110101";
            CLK_COR_SEQ_2_5 : bit_vector := "00010110101";
            ENMCOMMAALIGN => user_ENMCOMMAALIGN,
            ENPCOMMAALIGN => user_ENPCOMMAALIGN,
            LOOPBACK => user_LOOPBACK,
            POWERDOWN => user_POWERDOWN,
            REFCLK => user_REFCLK,
            REFCLK2 => user_REFCLK2,
            REFCLKSEL => user_REFCLKSEL,
            RXN => user_RXN,
            RXP => user_RXP,
            RXPOLARITY => user_RXPOLARITY,
            RXRESET => user_RXRESET,
            RXUSRCLK => user_RXUSRCLK,
            RXUSRCLK2 => user_RXUSRCLK2,
            TXBYPASS8B10B => user_TXBYPASS8B10B,
            TXCHARDISPMODE => user_TXCHARDISPMODE,
            TXCHARDISPVAL => user_TXCHARDISPVAL,
            TXCHARISK => user_TXCHARISK,
            TXDATA => user_TXDATA,
            TXFORCECRCERR => user_TXFORCECRCERR,
            TXINHIBIT => user_TXINHIBIT,
            TXPOLARITY => user_TXPOLARITY,
            TXRESET => user_TXRESET,
            TXUSRCLK => user_TXUSRCLK,
            TXUSRCLK2 => user_TXUSRCLK2);
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 4;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "1111111000";
CRC_END_OF_PKT : string := "K29_7";
CRC_FORMAT : string := "FIBRE_CHAN";
CRC_START_OF_PKT : string := "K27_7";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "1100000000";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "0011111000";
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 4;
RX_DECODE_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 4;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0;

-- synopsys translate_on
port {
  CONFIGOUT : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(3 downto 0);
  RXCHARISK : out std_logic_vector(3 downto 0);
  RXCHECKINGCRC : out std_logic_vector(3 downto 0);
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_ulogic;
  RXCRCERR : out std_ulogic;
  RXDATA : out std_logic_vector(31 downto 0);
  RXDISPERR : out std_logic_vector(3 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(3 downto 0);
  RXREALIGN : out std_ulogic;
  RXRECCCLK : out std_ulogic;
  RXRUNDISP : out std_logic_vector(3 downto 0);
  TXBUFERR : out std_ulogic;
  TXKERR : out std_logic_vector(3 downto 0);
  TXN : out std_ulogic;
  TXP : out std_ulogic;
  TXRUNDISP : out std_logic_vector(3 downto 0);
  BREFCLK : in std_ulogic := 'X';
  BREFCLK2 : in std_ulogic := 'X';
CONFIGENABLE        : in std_logic;
CONFIGIN           : in std_logic;
ENMCOMMAALIGN      : in std_logic;
ENPCOMMAALIGN      : in std_logic;
LOOPBACK           : in std_logic_vector(1 downto 0);
POWERDOWN          : in std_logic;
REFCLK             : in std_logic;
REFCLK2            : in std_logic;
REFCLKKSEL         : in std_logic;
RXN                : in std_logic;
RXP                : in std_logic;
RXPOLARITY         : in std_logic;
RXRESET            : in std_logic;
RXUSRCLK           : in std_logic;
RXUSRCLK2          : in std_logic;
TXBYPASS8B10B      : in std_logic_vector(3 downto 0);
TXCHARDISPMODE     : in std_logic_vector(3 downto 0);
TXCHARDISPVAL      : in std_logic_vector(3 downto 0);
TXCHARISK          : in std_logic_vector(3 downto 0);
TXDATA             : in std_logic_vector(31 downto 0);
TXFORCECRCERR      : in std_logic;
TXINHIBIT          : in std_logic;
TXPOLARITY         : in std_logic;
TXRESET            : in std_logic;
TXUSRCLK           : in std_logic;
TXUSRCLK2          : in std_logic);

end component;

-- Component Attribute specification for GT_FIBRE_CHAN_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_FIBRE_CHAN_4 should be placed
-- in architecture after the begin keyword

GT_FIBRE_CHAN_4_INSTANCE_NAME : GT_FIBRE_CHAN_4
-- synopsys translate_off
generic map (                 
    ALIGN_COMMA_MSB           => boolean_value,
    CHAN_BOND_LIMIT           => integer_value,
    CHAN_BOND_MODE            => string_value,
    CHAN_BOND_OFFSET          => integer_value,
    CHAN_BOND_ONE_SHOT        => boolean_value,
    CHAN_BOND_SEQ_1_1         => bit_value,
    CHAN_BOND_SEQ_1_2         => bit_value,
    CHAN_BOND_SEQ_1_3         => bit_value,
    CHAN_BOND_SEQ_1_4         => bit_value,
    CHAN_BOND_SEQ_2_1         => bit_value,
    CHAN_BOND_SEQ_2_2         => bit_value,
    CHAN_BOND_SEQ_2_3         => bit_value,
    CHAN_BOND_SEQ_2_4         => bit_value,
    CHAN_BOND_SEQ_2_USE       => boolean_value,
    CHAN_BOND_SEQ_LEN         => integer_value,
    CHAN_BOND_WAIT            => integer_value,
    CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLX_COR_KEEP_IDLE => boolean_value,
CLX_COR_REPEAT_WAIT => integer_value,
CLX_COR_SEQ_1_1 => bit_value,
CLX_COR_SEQ_1_2 => bit_value,
CLX_COR_SEQ_1_3 => bit_value,
CLX_COR_SEQ_1_4 => bit_value,
CLX_COR_SEQ_2_1 => bit_value,
CLX_COR_SEQ_2_2 => bit_value,
CLX_COR_SEQ_2_3 => bit_value,
CLX_COR_SEQ_2_4 => bit_value,
CLX_COR_SEQ_2_USE => boolean_value,
CLX_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (  
  CONFIGOUT => user_CONFIGOUT,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCHECKINGCRC => user_RXCHECKINGCRC,
  RCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXCRCERR => user_RXCRCERR,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCCLK => user_RXRECCCLK,
  RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => userPOWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESE T => user_RXRESE T,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

Verilog Instantiation Templates

GT_FIBRE_CHAN_1 Verilog Code

```
GT_FIBRE_CHAN_1 GT_FIBRE_CHAN_1_name(
   .CONFIGOUT (user_CONFIGOUT),
   .RXBUFSTATUS (user_RXBUFSTATUS),
   .RXCHARISCOMMA (user_RXCHARISCOMMA),
   .RXCHARISK (user_RXCHARISK),
   .RXCHECKINGCRC (user_RXCHECKINGCRC),
   .RXCLKCORCNT (user_RXCLKCORCNT),
   .RXCOMMADET (user_RXCOMMADET),
   .RXCRCERR (user_RXCRCERR),
   .RXDATA (user_RXDATA),
   .RXDISPERR (user_RXDISPERR),
   .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
   .RXNOTINTABLE (user_RXNOTINTABLE),
   .RXREALIGN (user_RXREALIGN),
   .RXRECCCLK (user_RXRECCCLK),
   .RXRUNDISP (user_RXRUNDISP),
   .TXBUFERR (user_TXBUFERR),
   .TXKERR (user_TXKERR),
   .TXN (user_TXN),
```
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLK (user_BREFCLK),
.BREFCLK2 (user_BREFCLK2),
.CONFIGENABLE (user_CONFIGENABLE),
.CONFIGIN (user_CONFIGIN),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPolarity),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),

.defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 1;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 0;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 7;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 2;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b001001110101;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b0001101101101;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00010110101101;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
GT_FIBRE_CHAN

GT_FIBRE_CHAN_2 Verilog Code

GT_FIBRE_CHAN_2 GT_FIBRE_CHAN_2_name(
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RCRCERR (user_RXCRCERR),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSOFSYNC (user_RXLOSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECCCLK (user_RXRECCCLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXERR (user_TXKERR),
    .TXN (user_TXN),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLK (user_BREFCLK),
    .BREFCLK2 (user_BREFCLK2),
);
.CONFIGENABLE (user_CONFIGENABLE),
.CONFIGIN (user_CONFIGIN),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.POWERDOWN (userPOWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 1;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 0;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 7;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 2;
defparam user_instance_name.CLK_CORSEQ_1_1 = 11'b00010111100;
defparam user_instance_name.CLK_CORSEQ_1_2 = 11'b00010111101;
defparam user_instance_name.CLK_CORSEQ_1_3 = 11'b00010111101;
defparam user_instance_name.CLK_CORSEQ_1_4 = 11'b00010111101;
defparam user_instance_name.CLK_CORSEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_CORSEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_CORSEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_CORSEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_CORSEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_CORSEQ_LEN = 4;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b111111100;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "FIBRE_CHAN";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMM_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 2;
defparam user_instance_name.RX_DECODE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INC = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 2;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

GT_FIBRE_CHAN_4 Verilog Code

GT_FIBRE_CHAN_4 name(
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXCRCERR (user_RXCRCERR),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLKL (user_RXRECLKL),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLK (user_BREFCLK),
    .REFCLK2 (user_BREFCLK2),
    .CONFIGENABLE (user_CONFIGENABLE),
    .CONFIGIN (user_CONFIGIN),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKSEL (user_REFCLKSEL),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASSB10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXFORCECRRCERR (user_TXFORCECRRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 1;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 0;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 7;
defparam user_instance_name.CLC_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLC_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLC_COR_REPEAT_WAIT = 2;
defparam user_instance_name.CLC_COR_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CLC_COR_SEQ_1_2 = 11'b00010101101;
defparam user_instance_name.CLC_COR_SEQ_1_3 = 11'b00010110101;
defparam user_instance_name.CLC_COR_SEQ_1_4 = 11'b00010110101;
defparam user_instance_name.CLC_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLC_COR_SEQ_LEN = 4;
defparam user_instance_name.CLC_COR_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b11111111100;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "FIBRE_CHAN";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
```vhdl
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 4;
defparam user_instance_name.RX_DECODE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 4;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;
```

Commonly Used Constraints

None
GT_INFINIBAND_n

Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT_INFINIBAND_n</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
<th>XC9500, XC9500XV, XC9500XL</th>
<th>CoolRunner XPLA3</th>
<th>CoolRunner-II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Primitive*</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

This Infiniband gigabit transceiver supports 1, 2, and 4-byte data paths. The letter $n$ represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of $n$. For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>CHBONDO(3:0)</td>
</tr>
<tr>
<td>CHBONDI(3:0)</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISCOMMA(0:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>RXCHARISK(0:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(7:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(0:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(0:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECCCLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(0:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(0:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(0:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(0:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPVAL(0:0)</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXCHARISK(0:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
### GT_INFINIBAND_2

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>CHBONDO(3:0)</td>
</tr>
<tr>
<td>CHBONDI(3:0)</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>RXCHARISK(1:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(15:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(1:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(1:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(1:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(1:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPVAL(1:0)</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
### GT_INFINIBAND_4

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLK2</td>
<td>CHBONDO(3:0)</td>
</tr>
<tr>
<td>CHBONDI(3:0)</td>
<td>CONFIGOUT</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
<td>RXBUFFSTATUS(1:0)</td>
</tr>
<tr>
<td>CONFIGIN</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>RXCHARISK(3:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>RXCHECKINGCRC</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCRCERR</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXDATA(31:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXLOSSOSYNC(1:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXNOTINTABLE(3:0)</td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(3:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXCHARDISPVAL(3:0)</td>
<td>TXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
<td></td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>

X9900
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT_INFINIBAND_1 VHDL Code

```vhdl
-- Component Attribute specification for GT_INFINIBAND_1
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT_INFINIBAND_1
  -- synopsys translate_off
  GENERIC(
    ALIGN_COMMA_MSB : BOOLEAN := FALSE;
    CHAN_BOND_LIMIT : INTEGER := 16;
    CHAN_BOND_MODE  : STRING := "OFF";
    CHAN_BOND_OFFSET : INTEGER := 8;
    CHAN_BOND_ONE_SHOT        BOOLEAN := FALSE;
    CHAN_BOND_SEQ_1_1        BIT_VECTOR := "00110111100";
    CHAN_BOND_SEQ_1_2        BIT_VECTOR := LANE_ID;
    CHAN_BOND_SEQ_1_3        BIT_VECTOR := "00001001100";
    CHAN_BOND_SEQ_1_4        BIT_VECTOR := "00001001100";
    CHAN_BOND_SEQ_2_1        BIT_VECTOR := "00110111100";
    CHAN_BOND_SEQ_2_2        BIT_VECTOR := LANE_ID;
    CHAN_BOND_SEQ_2_3        BIT_VECTOR := "00001001100";
    CHAN_BOND_SEQ_2_4        BIT_VECTOR := "00001001100";
    CHAN_BOND_SEQ_2_USE       BOOLEAN := TRUE;
    CHAN_BOND_SEQ_LEN      INTEGER := 4;
    CHAN_BOND_WAIT        INTEGER := 8;
    CLK_COR_INSERT_IDLE_FLAG BOOLEAN := FALSE;
    CLK_COR_KEEP_IDLE      BOOLEAN := FALSE;
    CLK_COR_REPEAT_WAIT   INTEGER := 1;
    CLK_COR_SEQ_1_1        BIT_VECTOR := "00100011100";
    CLK_COR_SEQ_1_2        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_1_3        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_1_4        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_2_1        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_2_2        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_2_3        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_2_4        BIT_VECTOR := "00000000000";
    CLK_COR_SEQ_2_USE       BOOLEAN := FALSE;
    CLK_COR_SEQ_LEN       INTEGER := 1;
    CLK_CORRECT_USE       BOOLEAN := TRUE;
    COMMA_10B_MASK        BIT_VECTOR := "1111111000";
    CRC_END_OF_PKT        STRING := "K29_7";
    CRC_FORMAT           STRING := "INFINIBAND";
    CRC_START_OF_PKT     STRING := "K27_7";
    DEC_MCOMMA_DETECT    BOOLEAN := TRUE;
    DEC_PCOMMA_DETECT    BOOLEAN := TRUE;
    DEC_VALID_COMMA_ONLY BOOLEAN := TRUE;
    LANE_ID              BIT_VECTOR := "00000000000";
    MCOMMA_10B_VALUE     BIT_VECTOR := "11000000000";
    MCOMMA_DETECT        BOOLEAN := TRUE;
    PCOMMA_10B_VALUE     BIT_VECTOR := "00111111100";
    PCOMMA_DETECT        BOOLEAN := TRUE;
  );
END COMPONENT;
```
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE  : boolean := TRUE;
RX_CRC_USE     : boolean := FALSE;
RX_DATA_WIDTH  : integer := 1;
RX_DCODE_USE   : boolean := TRUE;
RX_LOS_INVALID_INC : integer := 1;
RX_LOS_THRESHOLD: integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B     : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE  : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE     : boolean := FALSE;
TX_DATA_WIDTH  : integer := 1;
TX_DIFF_CTRL   : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port(
  CHBONDDONE : out std_ulogic;
  CHBONDO   : out std_logic_vector(3 downto 0);
  CONFIGOUT : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(0 downto 0);
  RXCHARISK  : out std_logic_vector(0 downto 0);
  RXCHECKINGCRC : out std_ulogic;
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_ulogic;
  RXCRCERR   : out std_ulogic;
  RXDATA     : out std_logic_vector(7 downto 0);
  RXDISERR   : out std_logic_vector(0 downto 0);
  RXLOSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(0 downto 0);
  RXREALIGN  : out std_ulogic;
  RXRECCLK   : out std_ulogic;
  RXRUNDISP  : out std_logic_vector(0 downto 0);
  TXBUFERR   : out std_ulogic;
  TXKERR     : out std_logic_vector(0 downto 0);
  TXN        : out std_ulogic;
  TXP        : out std_ulogic;
  TXRUNDISP  : out std_logic_vector(0 downto 0);
  BREFCLK    : in std_ulogic := 'X';
  BREFCLK2   : in std_ulogic := 'X';
  CHBOND1    : in std_logic_vector(3 downto 0);
  CONFIGENABLE : in std_ulogic;
  CONFIGIN   : in std_ulogic;
  ENCHANSYNC : in std_ulogic;
  ENMCOMMAALIGN : in std_ulogic;
  ENPCOMMAALIGN : in std_ulogic;
  LOOPBACK   : in std_logic_vector(1 downto 0);
  POWERDOWN  : in std_ulogic;
  REFCLK     : in std_ulogic;
  REFCLK2    : in std_ulogic;
  REFCLKSEL  : in std_ulogic;
  RXN        : in std_ulogic;
  RXP        : in std_ulogic;
  RXPOLARITY : in std_ulogic;
  RXRESET    : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASSBB10B : in std_logic_vector(0 downto 0);
TXCHARDISPMODE : in std_logic_vector(0 downto 0);
TXCHARDISPVAL : in std_logic_vector(0 downto 0);
TXCHARISK : in std_logic_vector(0 downto 0);
TXDATA : in std_logic_vector(7 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESEt : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic); 

end component;

-- Component Attribute specification for GT_INFINIBAND_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_INFINIBAND_1 should be placed
-- in architecture after the begin keyword

GT_INFINIBAND_1_INSTANCE_NAME : GT_INFINIBAND_1
-- synopsys translate_off
generic map ( 
ALIGN_COMMA_MSB => boolean_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_OFFSET => integer_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_WAIT => integer_value,
CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLK_COR_KEEP_IDLE => boolean_value,
CLK_COR_REPEAT_WAIT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
LANE_ID => bit_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (

CHBONDDONE => user_CHBONDDONE,
CHBONDO => user_CHBONDO,
CONFIGOUT => user_CONFIGOUT,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CHBONDI => user_CHBONDI,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLOCK,
REFCLOCK2 => user_REFCLOCK2,
REFCLINKSEL => user_REFCLINKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT_INFINIBAND_2 VHDL Code

-- Component Attribute specification for GT_INFINIBAND_2
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT_INFINIBAND_2
-- synopsys translate_off
generic (ALIGN_COMMA_MSB : boolean := FALSE;
CHAN_BOND_LIMIT : integer := 16;
CHAN_BOND_MODE : string := "OFF";
CHAN_BOND_OFFSET : integer := 8;
CHAN_BOND_ONE_SHOT : boolean := FALSE;
CHAN_BOND_SEQ_1_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_1_2 : bit_vector := LANE_ID;
CHAN_BOND_SEQ_1_3 : bit_vector := "00001001010";
CHAN_BOND_SEQ_1_4 : bit_vector := "00001001010";
CHAN_BOND_SEQ_2_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_2_2 : bit_vector := LANE_ID;
CHAN_BOND_SEQ_2_3 : bit_vector := "0001000101";
CHAN_BOND_SEQ_2_4 : bit_vector := "0001000101";
CHAN_BOND_SEQ_2_USE : boolean := TRUE;
CHAN_BOND_SEQ_LEN : integer := 4;
CHAN_BOND_WAIT : integer := 8;
CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
CLK_COR_KEEP_IDLE : boolean := FALSE;
CLK_COR_REPEAT_WAIT : integer := 1;
CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
);
CLK_COR_SEQ_2_1 : bit_vector := "000000000000";
CLK_COR_SEQ_2_2 : bit_vector := "000000000000";
CLK_COR_SEQ_2_3 : bit_vector := "000000000000";
CLK_COR_SEQ_2_4 : bit_vector := "000000000000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 1;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "111111100000";
CRC_END_OF_PKT : string := "K29_7";
CRC_FORMAT : string := "INFINIBAND";
CRC_START_OF_PKT : string := "K27_7";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
LANE_ID : bit_vector := "000000000000";
MCOMMA_10B_VALUE : bit_vector := "110000000000";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "001111100000";
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 2;
RX_DECODE_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 2;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port (}

CHBONDDONE : out std_ulogic;
CHBONO : out std_logic_vector(3 downto 0);
CONFIGOUT : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(1 downto 0);
RXCHARISK : out std_logic_vector(1 downto 0);
RXCHECKINGCRC : out std_ulogic;
RXCLKACORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXCRCERR : out std_ulogic;
RXDATA : out std_logic_vector(15 downto 0);
RXDISPERR : out std_logic_vector(1 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(1 downto 0);
RXREALIGN : out std_ulogic;
RXRECLCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(1 downto 0);
TXBUFERR : out std_ulogic;
TXXERR : out std_logic_vector(1 downto 0);
TXN : out std_ulogic;
end component;

-- Component Attribute specification for GT_INFINIBAND_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_INFINIBAND_2 should be placed
-- in architecture after the begin keyword

GT_INFINIBAND_2_INSTANCE_NAME : GT_INFINIBAND_2
-- synopsys translate_off
generic map (  
  ALIGN_COMMA_MSB           => boolean_value,  
  CHAN_BOND_LIMIT           => integer_value,  
  CHAN_BOND_MODE            => string_value,  
  CHAN_BOND_OFFSET          => integer_value,  
  CHAN_BOND_ONE_SHOT        => boolean_value,  
  CHAN_BOND_SEQ_1_1         => bit_value,  
  CHAN_BOND_SEQ_1_2         => bit_value,  
  CHAN_BOND_SEQ_1_3         => bit_value,  
  CHAN_BOND_SEQ_1_4         => bit_value,  
  CHAN_BOND_SEQ_2_1         => bit_value,  
  CHAN_BOND_SEQ_2_2         => bit_value,  
  TXBYPASS8B10B             => std_logic_vector(1 downto 0),  
  TXCHARDISPMODE            => std_logic_vector(1 downto 0),  
  TXCHARISPK                 => std_logic_vector(1 downto 0),  
  TXDATA                    => std_logic_vector(15 downto 0),  
  TXFORCECRCERR             => std_ulogic,  
  TXINHIBIT                 => std_ulogic,  
  TXPOLARITY                => std_ulogic,  
  TXRESET                   => std_ulogic,  
  TXUSRCLK                   => std_ulogic,  
  TXUSRCLK2                  => std_ulogic,  
  RXN                       => std_ulogic,  
  RXP                       => std_ulogic,  
  RXPOLARITY                => std_ulogic,  
  RXRESET                   => std_ulogic,  
  RXUSRCLK                  => std_ulogic,  
  RXUSRCLK2                  => std_ulogic,  
  BREFCLK                   => std_ulogic := 'X',  
  BREFCLK2                  => std_ulogic := 'X',  
  CHBOND1                   => std_logic_vector(3 downto 0),  
  CONFIGENABLE              => std_ulogic,  
  CONFIGIN                  => std_ulogic,  
  ENCHANSYNC                 => std_logic,  
  ENMCOMMAALIGN             => std_logic,  
  ENPCOMMAALIGN             => std_logic,  
  LOOPBACK                  => std_logic_vector(1 downto 0),  
  POWERDOWN                 => std_ulogic,  
  REFCLK                    => std_ulogic,  
  REFCLK2                   => std_ulogic,  
  REFCLKSEL                 => std_ulogic,  
  RXN                       => std_ulogic,  
  RXP                       => std_ulogic,  
  RXPOLARITY                => std_ulogic,  
  RXUSRCLK                  => std_ulogic,  
  RXUSRCLK2                  => std_ulogic);
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_WAIT => integer_value,
CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLK_COR_KEEP_IDLE => boolean_value,
CLK_COR_REPEAT_WAIT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
LANE_ID => bit_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value

-- synopsys translate_on
port map (
  CHBONDDONE => user_CHBONDDONE,
  CHBONDO => user_CHBONDO,
  CONFIGOUT => user_CONFIGOUT,
  RXBUFFSTATUS => user_RXBUFFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCHECKINGCRC => user_RXCHECKINGCRC,
  RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,  
RXCRCERR  => user_RXCRCERR,    
RXDATA    => user_RXDATA,      
RXDISPERR => user_RXDISPERR,   
RXLOSSOFSYNC => user_RXLOSSOFSYNC, 
RXNOTINTABLE => user_RXNOTINTABLE,  
RXREALIGN => user_RXREALIGN,  
RXRECCCLK => user_RXRECCCLK, 
RXRUNDISP => user_RXRUNDISP, 
TXBUFERR  => user_TXBUFERR, 
TXKERR    => user_TXKERR,     
TXN       => user_TXN,        
TXP       => user_TXP,        
TXRUNDISP => user_TXRUNDISP, 
BREFCLK   => user_BREFCLK,  
BREFCLK2  => user_BREFCLK2,  
CHBONDI   => user_CHBONDI,   
CONFIGENABLE => user_CONFIGENABLE,  
CONFIGIN => user_CONFIGIN,  
ENCHANSYNC => user_ENCHANSYNC,  
ENMCOMMAALIGN => user_ENMCOMMAALIGN,  
ENPCOMMAALIGN => user_ENPCOMMAALIGN,  
LOOPBACK => user_LOOPBACK,  
POWERDOWN => user_POWERDOWN,  
REFCLK    => user_REFCLK,  
REFCLK2   => user_REFCLK2,  
REFCLKSEL => user_REFCLKSEL, 
RXN       => user_RXN,        
RXP       => user_RXP,        
RXPOLARITY => user_RXPOLARITY,  
RXRESET   => user_RXRESET, 
RXUSRCLK  => user_RXUSRCLK,  
RXUSRCLK2 => user_RXUSRCLK2, 
TXBYPASS8B10B => user_TXBYPASS8B10B, 
TXCHARDISPMODE => user_TXCHARDISPMODE, 
TXCHARDISPVAL => user_TXCHARDISPVAL, 
TXCHARISK => user_TXCHARISK, 
TXDATA    => user_TXDATA,    
TXFORCECRCERR => user_TXFORCECRCERR, 
TXINHIBIT => user_TXINHIBIT, 
TXPOLARITY => user_TXPOLARITY, 
TXRESET   => user_TXRESET,  
TXUSRCLK  => user_TXUSRCLK,  
TXUSRCLK2 => user_TXUSRCLK2);

GT_INFINIBAND_4 VHDL Code

-- Component Attribute specification for GT_INFINIBAND_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_INFINIBAND_4
-- synopsys translate_off
generic (    
  ALIGN_COMMA_MSB         : boolean := FALSE;    
  CHAN_BOND_LIMIT         : integer := 16;    
  CHAN_BOND_MODE          : string := "OFF";   
  CHAN_BOND_OFFSET        : integer := 8;   
);
CHAN_BOND_ONE_SHOT : boolean := FALSE;
CHAN_BOND_SEQ_1_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_1_2 : bit_vector := LANE_ID;
CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_2_2 : bit_vector := LANE_ID;
CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_USE : boolean := TRUE;
CHAN_BOND_SEQ_LEN : integer := 4;
CHAN_BOND_WAIT : integer := 8;
CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
CLK_COR_KEEP_IDLE : boolean := FALSE;
CLK_COR_REPEAT_WAIT : integer := 1;
CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 1;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "1111111000";
CRC_END_OF_PKT : string := "K29_7";
CRC_FORMAT : string := "INFINIBAND";
CRC_START_OF_PKT : string := "K27_7";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
LANE_ID : bit_vector := "00000000000";
MCOMMA_10B_VALUE : bit_vector := "1100000000";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "0011111000";
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 4;
RX_DECODER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 4;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port (}
  CHBONDDONE : out std_ulogic;
end component;

-- Component Attribute specification for GT_INFINIBAND_4
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter constraints here

-- Component Instantiation for GT_INFINIBAND_4 should be placed
-- in architecture after the begin keyword

GT_INFINIBAND_4_INSTANCE_NAME : GT_INFINIBAND_4
-- synopsys translate_off

generic map (
  ALIGN_COMMA_MSB           => boolean_value,
  CHAN_BOND_LIMIT           => integer_value,
  CHAN_BOND_MODE            => string_value,
  CHAN_BOND_OFFSET          => integer_value,
  CHAN_BOND_ONE_SHOT        => boolean_value,
  CHAN_BOND_SEQ_1_1         => bit_value,
  CHAN_BOND_SEQ_1_2         => bit_value,
  CHAN_BOND_SEQ_1_3         => bit_value,
  CHAN_BOND_SEQ_1_4         => bit_value,
  CHAN_BOND_SEQ_2_1         => bit_value,
  CHAN_BOND_SEQ_2_2         => bit_value,
  CHAN_BOND_SEQ_2_3         => bit_value,
  CHAN_BOND_SEQ_2_4         => bit_value,
  CHAN_BOND_SEQ_2_USE       => boolean_value,
  CHAN_BOND_SEQ_LEN         => integer_value,
  CHAN_BOND_WAIT            => integer_value,
  CLK_COR_INSERT_IDLE_FLAG  => boolean_value,
  CLK_COR_KEEP_IDLE         => boolean_value,
  CLK_COR_REPEAT_WAIT       => integer_value,
  CLK_COR_SEQ_1_1           => bit_value,
  CLK_COR_SEQ_1_2           => bit_value,
  CLK_COR_SEQ_1_3           => bit_value,
  CLK_COR_SEQ_1_4           => bit_value,
  CLK_COR_SEQ_2_1           => bit_value,
  CLK_COR_SEQ_2_2           => bit_value,
  CLK_COR_SEQ_2_3           => bit_value,
  CLK_COR_SEQ_2_4           => bit_value,
  CLK_COR_SEQ_2_USE         => boolean_value,
  CLK_COR_SEQ_LEN           => integer_value,
  CLK_CORRECT_USE           => boolean_value,
  COMMA_10B_MASK            => bit_value,
  CRC_END_OF_PKT            => string_value,
  CRC_FORMAT                => string_value,
  CRC_START_OF_PKT          => string_value,
  DEC_MCOMMA_DETECT         => boolean_value,
  DEC_PCOMMA_DETECT         => boolean_value,
  DEC_VALID_COMMA_ONLY      => boolean_value,
  LANE_ID                   => bit_value,
  MCOMMA_10B_VALUE          => bit_value,
  MCOMMA_DETECT             => boolean_value,
  PCOMMA_10B_VALUE          => bit_value,
  PCOMMA_DETECT             => boolean_value,
  REF_CLK_V_SEL             => integer_value,
  RX_BUFFER_USE             => boolean_value,
  RX_CRC_USE                => boolean_value,
  RX_DATA_WIDTH             => integer_value,
  RX_DECODE_USE             => boolean_value,
  RX_LOS_INVALID_INCR       => integer_value,
  RX_LOS_THRESHOLD          => integer_value,
  RX_LOSS_OF_SYNC_FSM       => boolean_value,
SERDES_10B  => boolean_value,
TERMINATION_IMP  => integer_value,
TX_BUFFER_USE  => boolean_value,
TX_CRC_FORCE_VALUE  => bit_value,
TX_CRC_USE  => boolean_value,
TX_DATA_WIDTH  => integer_value,
TX_DIFF_CTRL  => integer_value,
TX_PREEMPHASIS  => integer_value)

-- synopsys translate_on
port map (
    CHBONDDONE  => user_CHBONDDONE,
    CHBONDO  => user_CHBONDO,
    CONFIGOUT  => user_CONFIGOUT,
    RXBUFSTATUS  => user_RXBUFSTATUS,
    RXCHARISCOMMA  => user_RXCHARISCOMMA,
    RXCHARISK  => user_RXCHARISK,
    RXCHECKINGCRC  => user_RXCHECKINGCRC,
    RXCLKCORCNT  => user_RXCLKCORCNT,
    RXCOMMADET  => user_RXCOMMADET,
    RXCRCERR  => user_RXCRCERR,
    RXDATA  => user_RXDATA,
    RXDISPERR  => user_RXDISPERR,
    RXLOSSOFSYNC  => user_RXLOSSOFSYNC,
    RXNOTINTABLE  => user_RXNOTINTABLE,
    RXREALIGN  => user_RXREALIGN,
    RXRECLK  => user_RXRECLK,
    RXRUNDISP  => user_RXRUNDISP,
    TXBUFERR  => user_TXBUFERR,
    TXKERR  => user_TXKERR,
    TXN  => user_TXN,
    TXP  => user_TXP,
    TXRUNDISP  => user_TXRUNDISP,
    BREFCLK  => user_BREFCLK,
    BREFCLK2  => user_BREFCLK2,
    CHBONDI  => user_CHBONDI,
    CONFIGENABLE  => user_CONFIGENABLE,
    CONFIGIN  => user_CONFIGIN,
    ENCHANSYNC  => user_ENCHANSYNC,
    ENMCOMMAALIGN  => user_ENMCOMMAALIGN,
    ENPCOMMAALIGN  => user_ENPCOMMAALIGN,
    LOOPBACK  => user_LOOPBACK,
    POWERDOWN  => user_POWERDOWN,
    REFCLK  => user_REFCLK,
    REFCLK2  => user_REFCLK2,
    REFCLKSEL  => user_REFCLKSEL,
    RN  => user_RXN,
    RP  => user_RXP,
    RXPOTAIRED    => user_RXPOLARITY,
    RXRES  => user_RXRESET,
    RXUSRCLK  => user_RXUSRCLK,
    RXUSRCLK2  => user_RXUSRCLK2,
    TXBYPASS8B10B  => user_TXBYPASS8B10B,
    TXCHARDISPMODE  => user_TXCHARDISPMODE,
    TXCHARDISPVAL  => user_TXCHARDISPVAL,
    TXCHARKS  => user_TXCHARKS,
    TXDATA  => user_TXDATA,
    TXFORCECRCERR  => user_TXFORCECRCERR,


```verilog
GT_INFINIBAND_1 GT_INFINIBAND_1_name(
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONO (user_CHBONO),
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADEXT (user_RXCOMMADEXT),
    .RXCRCERR (user_RXCRCERR),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLK (user_BREFCLK),
    .BREFCLK2 (user_BREFCLK2),
    .CHBONDI (user_CHBONDI),
    .CONFIGENABLE (user_CONFIGENABLE),
    .CONFIGIN (user_CONFIGIN),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .POWERDOWN (user_POWERDOWN),
    .REFCLK (user_REFCLK),
    .REFCLK2 (user_REFCLK2),
    .REFCLKSEL (user_REFCLKSEL),
    .RXN (user_RXN),
    .RXP (user_RXP),
    .RXPOLARITY (user_RXPOLARITY),
    .RXRESET (user_RXRESET),
    .RXUSRCLK (user_RXUSRCLK),
    .RXUSRCLK2 (user_RXUSRCLK2),
    .TXBYPASS8B10B (user_TXBYPASS8B10B),
    .TXCHARDISPMODE (user_TXCHARDISPMODE),
    .TXCHARDISPV (user_TXCHARDISPV),
    .TXCHARISK (user_TXCHARISK),
)
```

Verilog Instantiation Templates

**GT_INFINIBAND_1 Verilog Code**

- TXINHIBIT => user_TXINHIBIT,
- TXPOLARITY => user_TXPOLARITY,
- TXRESET => user_TXRESET,
- TXUSRCLK => user_TXUSRCLK,
- TXUSRCLK2 => user_TXUSRCLK2);
defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = LANE_ID;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = LANE_ID;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 4;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLC_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLC_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLC_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLC_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLC_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLC_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLC_COR_SEQ_LEN = 1;
defparam user_instance_name.CLC_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111100;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "INFINIBAND";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 1;
defparam user_instance_name.RX_DECODICE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOS_OF_SYNC_fsm = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 1;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

**GT_INFINIBAND_2 Verilog Code**

```verilog
GT_INFINIBAND_2 GT_INFINIBAND_2_name(
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .CONFIGOUT (user_CONFIGOUT),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCHECKINGCRC (user_RXCHECKINGCRC),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXCRCERR (user_RXCRCERR),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLK (user_BREFCLK),
    .BREFCLK2 (user_BREFCLK2),
    .CHBONDI (user_CHBONDI),
    .CONFIGENABLE (user_CONFIGENABLE),
    .CONFIGIN (user_CONFIGIN),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .POWERDOWN (user_POWERDOWN),
    .REFCLK (user_REFCLK),
    .REFCLK2 (user_REFCLK2),
    .REFCLKSEL (user_REFCLKSEL),
    .RXN (user_RXN),
    .RXP (user_RXP),
    .RXPOLARITY (user_RXPOLARITY),
    .RXRESET (user_RXRESET),
    .RXUSRCLK (user_RXUSRCLK),
    .RXUSRCLK2 (user_RXUSRCLK2),
    .TXBYPASS8B10B (user_TXBYPASS8B10B),
    .TXCHARDISPMODE (user_TXCHARDISPMODE),
    .TXCHARDISPVAL (user_TXCHARDISPVAL),
    .TXCHARISK (user_TXCHARISK),
    .TXDATA (user_TXDATA),
)
```
.TXFORCECRCERR (user_TXFORCECRCERR),
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = LANE_ID;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = LANE_ID;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 4;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111000;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "INFINIBAND";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 2;
defparam user_instance_name.RX_DECIMATE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 2;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

GT_INFINIBAND_4 Verilog Code

GT_INFINIBAND_4 GT_INFINIBAND_4.name(
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .CONFIGOUT (user_CONFIGOUT),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCHECKINGCRC (user_RXCHECKINGCRC),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXCRCERR (user_RXCRCERR),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECECLK (user_RXRECECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPV (user_TXCHARDISPV),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPV (user_TXCHARDISPV),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPV (user_TXCHARDISPV),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPV (user_TXCHARDISPV),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPV (user_TXCHARDISPV),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPV (user_TXCHARDISPV),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP))
)
.TXINHIBIT (user_TXINHIBIT),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b001110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = LANE_ID;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b001110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = LANE_ID;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 4;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 1;
defparam user_instance_name.CLK_CORRECTUSE = "TRUE";
defparam user_instance_name.COMMA_10B_VALUE = 10'b10001000100;
defparam user_instance_name.CRC_ENDIAN OF_PACKET = "K29_7";
defparam user_instance_name.CRC_FORMAT = "INFINIBAND";
defparam user_instance_name.CRC_START_OF_PACKET = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMM ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_Value = 10'b10001000100;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b10001000100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 4;
defparam user_instance_name.RX DECODE_USE = "TRUE";
defparam user_instance_name.RX LOS_INVALID_INCREMENT = 1;
defparam user_instance_name.RX LOS_THRESHOLD = 4;
defparam user_instance_name.RX LOSS_OF SYNC FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 4;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

Commonly Used Constraints
None
GT_XAUI\textsubscript{n}

Gigabit Transceiver for High-Speed I/O

Architectures Supported

| GT_XAUI\textsubscript{n} |     |
|--------------------------|--|--|
| Spartan-II, Spartan-IIE  | No |
| Spartan-3                | No |
| Virtex, Virtex-E         | Prinative* |
| Virtex-II, Virtex-II Pro, Virtex-II Pro X | No |
| XC9500, XC9500XV, XC9500XL | No |
| CoolRunner XPLA3         | No |
| CoolRunner-II            | No |

*Supported for Virtex-II Pro but not for Virtex-II or Virtex-II Pro X.

This XAUI gigabit transceiver supports 1, 2, and 4-byte data paths. The letter \(n\) represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \(n\). For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>GT_XAUI_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLK</td>
</tr>
<tr>
<td>BREFCLK2</td>
</tr>
<tr>
<td>CHBOND(3:0)</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
</tr>
<tr>
<td>CONFIGIN</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLKSSEL</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPVAL(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>GT_XAUI_4</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>BREFCLK</td>
</tr>
<tr>
<td>BREFCLK2</td>
</tr>
<tr>
<td>CHBONDI(3:0)</td>
</tr>
<tr>
<td>CONFIGENABLE</td>
</tr>
<tr>
<td>CONFIGIN</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLKSEL</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
</tr>
<tr>
<td>TXFORCECRCERR</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT_XAUI_1 VHDL Code

-- Component Attribute specification for GT_XAUI_1
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT_XAUI_1
-- synopsys translate_off
GENERIC (ALIGN_COMMA_MSB : BOOLEAN := FALSE;
CHAN_BOND_LIMIT : INTEGER := 16;
CHAN_BOND_MODE : STRING := "OFF";
CHAN_BOND_OFFSET : INTEGER := 8;
CHAN_BOND_ONE_SHOT : BOOLEAN := FALSE;
CHAN_BOND_SEQ_1_1 : BIT_VECTOR := "00101111100";
CHAN_BOND_SEQ_1_2 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_1_3 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_1_4 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_2_1 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_2_2 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_2_3 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_2_4 : BIT_VECTOR := "00000000000";
CHAN_BOND_SEQ_2_USE : BOOLEAN := FALSE;
CHAN_BOND_SEQ_LEN : INTEGER := 1;
CHAN_BOND_WAIT : INTEGER := 8;
CLK_COR_INSERT_IDLE_FLAG : BOOLEAN := FALSE;
CLK_COR_KEEP_IDLE : BOOLEAN := FALSE;
CLK_COR_REPEAT_WAIT : INTEGER := 1;
CLK_COR_SEQ_1_1 : BIT_VECTOR := "00100011100";
CLK_COR_SEQ_1_2 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_1_3 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_1_4 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_2_1 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_2_2 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_2_3 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_2_4 : BIT_VECTOR := "00000000000";
CLK_COR_SEQ_2_USE : BOOLEAN := FALSE;
CLK_COR_SEQ_LEN : INTEGER := 1;
CLK_CORRECT_USE : BOOLEAN := TRUE;
COMMA_10B_MASK : BIT_VECTOR := "11111111000";
CRC_END_OF_PKT : STRING := "K29_7";
CRC_FORMAT : STRING := "USER_MODE";
CRC_START_OF_PKT : STRING := "K27_7";
DEC_MCOMMA_DETECT : BOOLEAN := TRUE;
DEC_PCOMMA_DETECT : BOOLEAN := TRUE;
DEC_VALID_COMMA_ONLY : BOOLEAN := TRUE;
MCOMMA_10B_VALUE : BIT_VECTOR := "11000000000";
MCOMMA_DETECT : BOOLEAN := TRUE;
PCOMMA_10B_VALUE : BIT_VECTOR := "00111111000";
PCOMMA_DETECT : BOOLEAN := TRUE;
REF_CLK_V_SEL : INTEGER := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 1;
RX_DECODING_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOSS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "11010110";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 1;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port (  
  CHBONDDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(3 downto 0);
  CONFIGOUT : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(0 downto 0);
  RXCHARISK : out std_logic_vector(0 downto 0);
  RXCHECKINGCRC : out std_logic;
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_ulogic;
  RXCRCERR : out std_ulogic;
  RXDATA : out std_logic_vector(7 downto 0);
  RXDISPERR : out std_logic_vector(0 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(0 downto 0);
  RXREALIGN : out std_ulogic;
  RXRECCLK : out std_ulogic;
  RXRUNDISP : out std_logic_vector(0 downto 0);
  TXBUFERR : out std_logic;
  TXKERR : out std_logic_vector(0 downto 0);
  TXN : out std_ulogic;
  TXP : out std_ulogic;
  TXRUNDISP : out std_logic_vector(0 downto 0);
  BREFCLK : in std_ulogic := 'X';
  BREFCLK2 : in std_ulogic := 'X';
  CHBOND1 : in std_logic_vector(3 downto 0);
  CONFIGENABLE : in std_ulogic;
  CONFIGIN : in std_ulogic;
  ENCHANSYNC : in std_ulogic;
  ENMCOMMAALIGN : in std_ulogic;
  ENPCOMMAALIGN : in std_ulogic;
  LOOPBACK : in std_logic_vector(1 downto 0);
  POWERDOWN : in std_ulogic;
  REFCLK : in std_ulogic;
  REFCLK2 : in std_ulogic;
  REFCLKSEL : in std_ulogic;
  RXN : in std_ulogic;
  RXP : in std_ulogic;
  RXPOLARITY : in std_ulogic;
  RXRESET : in std_ulogic;
  RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(0 downto 0);
TXCHARDISPMODE : in std_logic_vector(0 downto 0);
TXCHARDISPVAL : in std_logic_vector(0 downto 0);
TXCHARISK : in std_logic_vector(0 downto 0);
TXDATA : in std_logic_vector(7 downto 0);
TXFORCECRCERR : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT_XAUI_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_XAUI_1 should be placed
-- in architecture after the begin keyword

GT_XAUI_1_INSTANCE_NAME : GT_XAUI_1
-- synopsys translate_off
generic map (ALIGN_COMMA_MSB => boolean_value, CHAN_BOND_LIMIT => integer_value, CHAN_BOND_MODE => string_value, CHAN_BOND_OFFSET => integer_value, CHAN_BOND_ONE_SHOT => boolean_value, CHAN_BOND_SEQ_1_1 => bit_value, CHAN_BOND_SEQ_1_2 => bit_value, CHAN_BOND_SEQ_1_3 => bit_value, CHAN_BOND_SEQ_1_4 => bit_value, CHAN_BOND_SEQ_2_1 => bit_value, CHAN_BOND_SEQ_2_2 => bit_value, CHAN_BOND_SEQ_2_3 => bit_value, CHAN_BOND_SEQ_2_4 => bit_value, CHAN_BOND_SEQ_2_USE => boolean_value, CHAN_BOND_SEQ_LEN => integer_value, CHAN_BOND_WAIT => integer_value, CLK_COR_INSERT_IDLE_FLAG => boolean_value, CLK_COR_KEEP_IDLE => boolean_value, CLK_COR_REPEAT_WAIT => integer_value, CLK_COR_SEQ_1_1 => bit_value, CLK_COR_SEQ_1_2 => bit_value, CLK_COR_SEQ_1_3 => bit_value, CLK_COR_SEQ_1_4 => bit_value, CLK_COR_SEQ_2_1 => bit_value, CLK_COR_SEQ_2_2 => bit_value, CLK_COR_SEQ_2_3 => bit_value, CLK_COR_SEQ_2_4 => bit_value, CLK_COR_SEQ_2_USE => boolean_value, CLK_COR_SEQ_LEN => integer_value, CLK_CORRECT_USE => boolean_value, COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECODE_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value

-- synopsys translate_on
port map (CHBONDDONE => user_CHBONDDONE,
CHBONDO => user_CHBONDO,
CONFIGOUT => user_CONFIGOUT,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISK => user_RXCHARISK,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHECKINGCRC => user_RXCHECKINGCRC,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXCRCERR => user_RXCRCERR,
RXDATA => user_RXDATA,
RXDISERR => user_RXDISERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCLK => user_RXRECCLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CHBONDI => user_CHBONDI,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT_XAUI_2 VHDL Code

-- Component Attribute specification for GT_XAUI_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_XAUI_2
-- synopsys translate_off
generic (}
    ALIGN_COMMA_MSB : boolean := FALSE;
    CHAN_BOND_LIMIT : integer := 16;
    CHAN_BOND_MODE : string := "OFF";
    CHAN_BOND_OFFSET : integer := 8;
    CHAN_BOND_ONE_SHOT : boolean := FALSE;
    CHAN_BOND_SEQ_1_1 : bit_vector := "00101111100";
    CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
    CHAN_BOND_SEQ_LEN : integer := 1;
    CHAN_BOND_WAIT : integer := 8;
    CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
    CLK_COR_KEEP_IDLE : boolean := FALSE;
    CLK_COR_REPEAT_WAIT : integer := 1;
    CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
    CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
    CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
    CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
    CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
    CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
-- synopsys translate_on
port (  
  CHBONDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(3 downto 0);
  CONFIGOUT : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(1 downto 0);
  RXCHARISK : out std_logic_vector(1 downto 0);
  RXCHECKINGCRC : out std_ulogic;
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_uLogic;
  RXCRCERR : out std_uLogic;
  RXDATA : out std_logic_vector(15 downto 0);
  RXDISPFERR : out std_logic_vector(1 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(1 downto 0);
  RXREALIGN : out std_uLogic;
  RXRECLK : out std_uLogic;
  RXRUNDISP : out std_logic_vector(1 downto 0);
  TXBUFERR : out std_uLogic;
  TXKERR : out std_logic_vector(1 downto 0);
  TXN : out std_uLogic;
  TXP : out std_uLogic;
  TXRUNDISP : out std_logic_vector(1 downto 0);
  BREFCLK : in std_uLogic := 'X';

  CLK_COR_SEQ_2_3 : bit_vector := "000000000000";
  CLK_COR_SEQ_2_4 : bit_vector := "000000000000";
  CLK_COR_SEQ_2_USE : boolean := FALSE;
  CLK_COR_SEQ_LEN : integer := 1;
  CLK_CORRECT_USE : boolean := TRUE;
  COMMA_10B_MASK : bit_vector := "1111111000";
  CRC_END_OF_PKT : string := "K29_7";
  CRC_FORMAT : string := "USER_MODE";
  CRC_START_OF_PKT : string := "K27_7";
  DEC_MCOMMA_DETECT : boolean := TRUE;
  DEC_PCOMMA_DETECT : boolean := TRUE;
  DEC_VALID_COMMA_ONLY : boolean := TRUE;
  MCOMMA_10B_VALUE : bit_vector := "11000000000";
  MCOMMA_DETECT : boolean := TRUE;
  PCOMMA_10B_VALUE : bit_vector := "0011111000";
  PCOMMA_DETECT : boolean := TRUE;
  REF_CLK_V_SEL : integer := 0;
  RX_BUFFER_USE : boolean := TRUE;
  RX_CRC_USE : boolean := FALSE;
  RX_DATA_WIDTH : integer := 2;
  RX_DECIMATE_USE : boolean := TRUE;
  RX_LOS_INVALID_INCR : integer := 1;
  RX_LOS_THRESHOLD : integer := 4;
  RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
  SERDES_10B : boolean := FALSE;
  TERMINATION_IMP : integer := 50;
  TX_BUFFER_USE : boolean := TRUE;
  TX_CRC_FORCE_VALUE : bit_vector := "11010110";
  TX_CRC_USE : boolean := FALSE;
  TX_DATA_WIDTH : integer := 2;
  TX_DIFF_CTRL : integer := 500;
  TX_PREEMPHASIS : integer := 0);
end component;

-- Component Attribute specification for GT_XAUI_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_XAUI_2 should be placed
-- in architecture after the begin keyword

GT_XAUI_2_INSTANCE_NAME : GT_XAUI_2
-- synopsys translate_off
generic map (
  ALIGN_COMMA_MSB => boolean_value,
  CHAN_BOND_LIMIT => integer_value,
  CHAN_BOND_MODE => string_value,
  CHAN_BOND_OFFSET => integer_value,
  CHAN_BOND_ONE_SHOT => boolean_value,
  CHAN_BOND_SEQ_1_1 => bit_value,
  CHAN_BOND_SEQ_1_2 => bit_value,
  CHAN_BOND_SEQ_1_3 => bit_value,
  CHAN_BOND_SEQ_1_4 => bit_value,
  CHAN_BOND_SEQ_2_1 => bit_value,
  CHAN_BOND_SEQ_2_2 => bit_value,
  CHAN_BOND_SEQ_2_3 => bit_value,
  CHAN_BOND_SEQ_2_4 => bit_value,
  CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_WAIT => integer_value,
CLK_COR_INSERT_IDLE_FLAG => boolean_value,
CLK_COR_KEEP_IDLE => boolean_value,
CLK_COR_REPEAT_WAIT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
CRC_END_OF_PKT => string_value,
CRC_FORMAT => string_value,
CRC_START_OF_PKT => string_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
REF_CLK_V_SEL => integer_value,
RX_BUFFER_USE => boolean_value,
RX_CRC_USE => boolean_value,
RX_DATA_WIDTH => integer_value,
RX_DECORUSE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SERDES_10B => boolean_value,
TERMINATION_IMP => integer_value,
TX_BUFFER_USE => boolean_value,
TX_CRC_FORCE_VALUE => bit_value,
TX_CRC_USE => boolean_value,
TX_DATA_WIDTH => integer_value,
TX_DIFF_CTRL => integer_value,
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (
  CHBONDONE => user_CHBONDONE,
  CHBONO => user_CHBONO,
  CONFIGOUT => user_CONFIGOUT,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCHECKINGCRC => user_RXCHECKINGCRC,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXCRCERR => user_RXCRCERR,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCLK => user_RXRECCLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXERR => user_TXERR,
TXN => user_TXN,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLK => user_BREFCLK,
BREFCLK2 => user_BREFCLK2,
CHBOND1 => user_CHBOND1,
CONFIGENABLE => user_CONFIGENABLE,
CONFIGIN => user_CONFIGIN,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKSEL => user_REFCLKSEL,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXFORCECRCERR => user_TXFORCECRCERR,
TXINHIBIT => user_TXINHIBIT,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT_XAUI_4 VHDL Code

-- Component Attribute specification for GT_XAUI_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT_XAUI_4
-- synopsys translate_off
generic(
   ALIGN_COMMA_MSB : boolean := FALSE;
   CHAN_BOND_LIMIT : integer := 16;
   CHAN_BOND_MODE : string := "OFF";
   CHAN_BOND_OFFSET : integer := 8;
   CHAN_BOND_ONE_SHOT : boolean := FALSE;
   CHAN_BOND_SEQ_1_1 : bit_vector := "00101111100";
   CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
   );
CHAN_BOND_SEQ_1_4 : bit_vector := "0000000000000000";
CHAN_BOND_SEQ_2_1 : bit_vector := "0000000000000000";
CHAN_BOND_SEQ_2_2 : bit_vector := "0000000000000000";
CHAN_BOND_SEQ_2_3 : bit_vector := "0000000000000000";
CHAN_BOND_SEQ_2_4 : bit_vector := "0000000000000000";
CHAN_BOND_SEQ_2_USE : boolean := FALSE;
CHAN_BOND_SEQ_LEN : integer := 1;
CHAN_BOND_WAIT : integer := 8;
CLK_COR_INSERT_IDLE_FLAG : boolean := FALSE;
CLK_COR_KEEP_IDLE : boolean := FALSE;
CLK_COR_REPEATE_WAIT : integer := 1;
CLK_COR_SEQ_1_1 : bit_vector := "0010001110000000";
CLK_COR_SEQ_1_2 : bit_vector := "0000000000000000";
CLK_COR_SEQ_1_3 : bit_vector := "0000000000000000";
CLK_COR_SEQ_1_4 : bit_vector := "0000000000000000";
CLK_COR_SEQ_2_1 : bit_vector := "0000000000000000";
CLK_COR_SEQ_2_2 : bit_vector := "0000000000000000";
CLK_COR_SEQ_2_3 : bit_vector := "0000000000000000";
CLK_COR_SEQ_2_4 : bit_vector := "0000000000000000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 1;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "1111111000000000";
CRC_END_OF_PKT : string := "K29_7";
CRC_FORMAT : string := "USER_MODE";
CRC_START_OF_PKT : string := "K27_7";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "1100000000000000";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "0011111000000000";
PCOMMA_DETECT : boolean := TRUE;
REF_CLK_V_SEL : integer := 0;
RX_BUFFER_USE : boolean := TRUE;
RX_CRC_USE : boolean := FALSE;
RX_DATA_WIDTH : integer := 4;
RX_DECODE_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SERDES_10B : boolean := FALSE;
TERMINATION_IMP : integer := 50;
TX_BUFFER_USE : boolean := TRUE;
TX_CRC_FORCE_VALUE : bit_vector := "1101011000000000";
TX_CRC_USE : boolean := FALSE;
TX_DATA_WIDTH : integer := 4;
TX_DIFF_CTRL : integer := 500;
TX_PREEMPHASIS : integer := 0);

-- synopsys translate_on
port (}
CHBONDDONE : out std_ulogic;
CHBONDO : out std_logic_vector(3 downto 0);
CONFIGOUT : out std_ulogic;
RXBUFSIZE : out std_logic_vector(1 downto 0);
RXCHARISCO : out std_logic_vector(3 downto 0);
RXCHARISK : out std_logic_vector(3 downto 0);
end component;

-- Component Attribute specification for GT_XAUI_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT_XAUI_4 should be placed
-- in architecture after the begin keyword
GT_XAUI_4_INSTANCE_NAME : GT_XAUI_4
-- synopsys translate_off

generic map (   
  ALIGN_COMMA_MSB => boolean_value,   
  CHAN_BOND_LIMIT => integer_value,   
  CHAN_BOND_MODE => string_value,   
  CHAN_BOND_OFFSET => integer_value,   
  CHAN_BOND_ONE_SHOT => boolean_value,   
  CHAN_BOND_SEQ_1_1 => bit_value,   
  CHAN_BOND_SEQ_1_2 => bit_value,   
  CHAN_BOND_SEQ_1_3 => bit_value,   
  CHAN_BOND_SEQ_1_4 => bit_value,   
  CHAN_BOND_SEQ_2_1 => bit_value,   
  CHAN_BOND_SEQ_2_2 => bit_value,   
  CHAN_BOND_SEQ_2_3 => bit_value,   
  CHAN_BOND_SEQ_2_4 => bit_value,   
  CHAN_BOND_SEQ_2_USE => boolean_value,   
  CHAN_BOND_SEQ_LEN => integer_value,   
  CHAN_BOND_WAIT => integer_value,   
  CLK_COR_INSERT_IDLE_FLAG => boolean_value,   
  CLK_COR_KEEP_IDLE => boolean_value,   
  CLK_COR_REPEATEWAIT => integer_value,   
  CLK_COR_SEQ_1_1 => bit_value,   
  CLK_COR_SEQ_1_2 => bit_value,   
  CLK_COR_SEQ_1_3 => bit_value,   
  CLK_COR_SEQ_1_4 => bit_value,   
  CLK_COR_SEQ_2_1 => bit_value,   
  CLK_COR_SEQ_2_2 => bit_value,   
  CLK_COR_SEQ_2_3 => bit_value,   
  CLK_COR_SEQ_2_4 => bit_value,   
  CLK_COR_SEQ_2_USE => boolean_value,   
  CLK_COR_SEQ_LEN => integer_value,   
  CLK_CORRECT_USE => boolean_value,   
  COMMA_10B_MASK => bit_value,   
  CRC_END_OF_PKT => string_value,   
  CRC_FORMAT => string_value,   
  CRC_START_OF_PKT => string_value,   
  DEC_MCOMMA_DETECT => boolean_value,   
  DEC_PCOMMA_DETECT => boolean_value,   
  DEC_VALID_COMMA_ONLY => boolean_value,   
  MCOMMA_10B_VALUE => bit_value,   
  MCOMMA_DETECT => boolean_value,   
  PCOMMA_10B_VALUE => bit_value,   
  PCOMMA_DETECT => boolean_value,   
  REF_CLK_V_SEL => integer_value,   
  RX_BUFFER_USE => boolean_value,   
  RX_CRC_USE => boolean_value,   
  RX_DATA_WIDTH => integer_value,   
  RX_DECODE_USE => boolean_value,   
  RX_LOS_INVALID_INCR => integer_value,   
  RX_LOS_THRESHOLD => integer_value,   
  RX_LOSS_OF_SYNC_FSM => boolean_value,   
  SERDES_10B => boolean_value,   
  TERMINATION_IMP => integer_value,   
  TX_BUFFER_USE => boolean_value,   
  TX_CRC_FORCE_VALUE => bit_value,   
  TX_CRC_USE => boolean_value,   
  TX_DATA_WIDTH => integer_value,   
)
TX_DIFF_CTRL => integer_value,  
TX_PREEMPHASIS => integer_value)

-- synopsys translate_on
port map (  
CHBONDDONE => user_CHBONDDONE,  
CHBOND => user_CHBOND,  
CONFIGOUT => user_CONFIGOUT,  
RXBUFSTATUS => user_RXBUFSTATUS,  
RXCHARISCOMMA => user_RXCHARISCOMMA,  
RXCHARISK => user_RXCHARISK,  
RXCHECKINGCRC => user_RXCHECKINGCRC,  
RXCLKCORCNT => user_RXCLKCORCNT,  
RXCOMMADET => user_RXCOMMADET,  
RXCRCERR => user_RXCRCERR,  
RXDATA => user_RXDATA,  
RXDISPERR => user_RXDISPERR,  
RXLOSSOFSYNC => user_RXLOSSOFSYNC,  
RXNOTINTABLE => user_RXNOTINTABLE,  
RXREALIGN => user_RXREALIGN,  
RXRECLK => user_RXRECLK,  
RXRUNDISP => user_RXRUNDISP,  
TXBUFERR => user_TXBUFERR,  
TXKERR => user_TXKERR,  
TXN => user_TXN,  
TXP => user_TXP,  
TXRUNDISP => user_TXRUNDISP,  
BREFCLK => user_BREFCLK,  
BREFCLK2 => user_BREFCLK2,  
CHBONDI => user_CHBONDI,  
CONFIGENABLE => user_CONFIGENABLE,  
CONFIGIN => user_CONFIGIN,  
ENCHANSYNC => user_ENCHANSYNC,  
ENMCOMMAALIGN => user_ENMCOMMAALIGN,  
ENPCOMMAALIGN => user_ENPCOMMAALIGN,  
LOOPBACK => user_LOOPBACK,  
POWERDOWN => user_POWERDOWN,  
REFCLK => user_REFCLK,  
REFCLK2 => user_REFCLK2,  
REFCLKSEL => user_REFCLKSEL,  
RXN => user_RXN,  
RXP => user_RXP,  
RXPOLARITY => user_RXPOLARITY,  
RXRESET => user_RXRESET,  
RXUSRCLK => user_RXUSRCLK,  
RXUSRCLK2 => user_RXUSRCLK2,  
TXBYPASS8B10B => user_TXBYPASS8B10B,  
TXCHARDISPMODE => user_TXCHARDISPMODE,  
TXCHARDISPVAL => user_TXCHARDISPVAL,  
TXCHARISK => user_TXCHARISK,  
TXDATA => user_TXDATA,  
TXFORCECRCERR => user_TXFORCECRCERR,  
TXINHIBIT => user_TXINHIBIT,  
TXPOLARITY => user_TXPOLARITY,  
TXRESETH => user_TXRESETH,  
TXUSRCLK => user_TXUSRCLK,  
TXUSRCLK2 => user_TXUSRCLK2);
Verilog Instantiation Templates

GT_XAUI_1 Verilog Code

GT_XAUI_1 GT_XAUI_1_name(
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .CONFIGOUT (user_CONFIGOUT),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCHECKINGCRC (user_RXCHECKINGCRC),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXCRCERR (user_RXCRCERR),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLK (user_BREFCLK),
  .BREFCLK2 (user_BREFCLK2),
  .CHBONDI (user_CHBONDI),
  .CONFIGENABLE (user_CONFIGENABLE),
  .CONFIGIN (user_CONFIGIN),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKSEL (user_REFCLKSEL),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPVAL (user_TXCHARDISPVAL),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXFORCECRCERR (user_TXFORCECRCERR),
  .TXINHIBIT (user_TXINHIBIT),
  .TXPOLARITY (user_TXPOLARITY),
  .TXRESET (user_TXRESET),
  .TXUSRCLK (user_TXUSRCLK),

.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 1;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111000;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 1;
defparam user_instance_name.RX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;

GT_XAUI_2 Verilog Code

GT_XAUI_2 GT_XAUI_2_name(
   .CHBONDDONE (user_CHBONDDONE),
   .CHBONDO (user_CHBONDO),
   .CONFIGOUT (user_CONFIGOUT),
   .RXBUFSTATUS (user_RXBUFSTATUS),
   .RXCHARISCOMMA (user_RXCHARISCOMMA),
   .RXCHARISK (user_RXCHARISK),
   .RXCHECKINGCRC (user_RXCHECKINGCRC),
   .RXCLKCORCNT (user_RXCLKCORCNT),
   .RXCOMMADET (user_RXCOMMADET),
   .RXCRCERR (user_RXCRCERR),
   .RXDATA (user_RXDATA),
   .RXDISPERR (user_RXDISPERR),
   .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
   .RXNOTINTABLE (user_RXNOTINTABLE),
   .RXREALIGN (user_RXREALIGN),
   .RXRECLK (user_RXRECLK),
   .RXRUNDISP (user_RXRUNDISP),
   .TXBUFERR (user_TXBUFERR),
   .TXKERR (user_TXKERR),
   .TXN (user_TXN),
   .TXP (user_TXP),
   .TXRUNDISP (user_TXRUNDISP),
   .BREFCLK (user_BREFCLK),
   .BREFCLK2 (user_BREFCLK2),
   .CHBOND (user_CHBOND),
   .CONFIGENABLE (user_CONFIGENABLE),
   .CONFIGIN (user_CONFIGIN),
   .ENCHANSYNC (user_ENCHANSYNC),
   .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
   .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
   .LOOPBACK (user_LOOPBACK),
   .POWERDOWN (user_POWERDOWN),
   .REFCLK (user_REFCLK),
   .REFCLK2 (user_REFCLK2),
   .REFCLKSEL (user_REFCLKSEL),
   .RXN (user_RXN),
   .RXP (user_RXP),
   .RXPOLARITY (user_RXPOLARITY),
   .RXRESET (user_RXRESET),
   .RXUSRCLK (user_RXUSRCLK),
   .RXUSRCLK2 (user_RXUSRCLK2),
   .TXBYPASS8B10B (user_TXBYPASS8B10B),
   .TXCHARDISPMODE (user_TXCHARDISPMODE),
   .TXCHARDISPMODE (user_TXCHARDISPMODE),
   .TXCHARDISPMODE (user_TXCHARDISPMODE),
   .TXCHARDISPMODE (user_TXCHARDISPMODE),
   .TXDATA (user_TXDATA),
   .TXFORCECRCERR (user_TXFORCECRCERR),
   .TXINHIBIT (user_TXINHIBIT),
   .TXP (user_TXP),
   .TXP (user_TXP),
   .TXRUNDISP (user_TXRUNDISP),
   .TXUSRCLK (user_TXUSRCLK),
   .TXUSRCLK2 (user_TXUSRCLK2));
defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_LEN = 1;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_2_LEN = 1;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111100;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 2;
defparam user_instance_name.RX_DECODE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RXLOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 2;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;
GT_XAUI_4 Verilog Code

GT_XAUI_4_GT_XAUI_4_name(
   .CHBONDDONE (user_CHBONDDONE),
   .CHBONDO (user_CHBONDO),
   .CONFIGOUT (user_CONFIGOUT),
   .RXBUFSTATUS (user_RXBUFSTATUS),
   .RXCHARISCOMMA (user_RXCHARISCOMMA),
   .RXCHARISK (user_RXCHARISK),
   .RXCHECKINGCRC (user_RXCHECKINGCRC),
   .RXCLKCORCNT (user_RXCLKCORCNT),
   .RXCOMMADET (user_RXCOMMADET),
   .RXCRCERR (user_RXCRCERR),
   .RXDATA (user_RXDATA),
   .RXDISPERR (user_RXDISPERR),
   .RXLOSOFSYNC (user_RXLOSOFSYNC),
   .RXNOTINTABLE (user_RXNOTINTABLE),
   .RXREALIGN (user_RXREALIGN),
   .RXRECECLK (user_RXRECECLK),
   .RXRUNDFSP (user_RXRUNDFSP),
   .TXBUFERR (user_TXBUFERR),
   .TXKERR (user_TXKERR),
   .TXN (user_TXN),
   .TXP (user_TXP),
   .TXRUNDFSP (user_TXRUNDFSP),

   .BREFCLK (user_BREFCLK),
   .BREFCLK2 (user_BREFCLK2),
   .CHBONDI (user_CHBONDI),
   .CONFIGENABLE (user_CONFIGENABLE),
   .CONFIGIN (user_CONFIGIN),
   .ENCHANSYNC (user_ENCHANSYNC),
   .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
   .ENPOMCOMMAALIGN (user_ENPOMCOMMAALIGN),
   .LOOPBACK (user_LOOPBACK),
   .POWERDOWN (user_POWERDOWN),
   .REFCLK (user_REFLCLK),
   .REFCLK2 (user_REFLCLK2),
   .REFCLKSEL (user_REFLCLKSEL),
   .RXN (user_RXN),
   .RXP (user_RXP),
   .RXPOLARITY (user_RXPOLARITY),
   .RXRESET (user_RXRESET),
   .RXUSRCCLK (user_RXUSRCCLK),
   .RXUSRCCLK2 (user_RXUSRCCLK2),
   .TXBYPASS8B10B (user_TXBYPASS8B10B),
   .TXCHARDISP (user_TXCHARDISP),
   .TXCHARDISPVAL (user_TXCHARDISPVAL),
   .TXCHARISK (user_TXCHARISK),
   .TXDATA (user_TXDATA),
   .TXFORCECRCERR (user_TXFORCECRCERR),
   .TXINHIBIT (user_TXINHIBIT),
   .TXPOLARITY (user_TXPOLARITY),
   .TXRESET (user_TXRESET),
   .TXUSRCCLK (user_TXUSRCCLK),
   .TXUSRCCLK2 (user_TXUSRCCLK2));

   defparam user_instance_name.ALIGN_COMMA_MSB = "FALSE";
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_OFFSET = 8;
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_WAIT = 8;
defparam user_instance_name.CLK_COR_INSERT_IDLE_FLAG = "FALSE";
defparam user_instance_name.CLK_COR_KEEP_IDLE = "FALSE";
defparam user_instance_name.CLK_COR_REPEAT_WAIT = 1;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 1;
defparam user_instance_name.CLK_COR_RECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b1111111100;
defparam user_instance_name.CRC_END_OF_PKT = "K29_7";
defparam user_instance_name.CRC_FORMAT = "USER_MODE";
defparam user_instance_name.CRC_START_OF_PKT = "K27_7";
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1100000000;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0011111000;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.REF_CLK_V_SEL = 0;
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.RX_CRC_USE = "FALSE";
defparam user_instance_name.RX_DATA_WIDTH = 4;
defparam user_instance_name.RX_DECODE_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SERDES_10B = "FALSE";
defparam user_instance_name.TERMINATION_IMP = 50;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_CRC_FORCE_VALUE = 8'b11010110;
defparam user_instance_name.TX_CRC_USE = "FALSE";
defparam user_instance_name.TX_DATA_WIDTH = 4;
defparam user_instance_name.TX_DIFF_CTRL = 500;
defparam user_instance_name.TX_PREEMPHASIS = 0;
Commonly Used Constraints

None
10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_AURORA_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Xilinx protocol 10-gigabit transceiver supports 1, 2, and 4-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 1, 2, or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>GT10_AURORA_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRFCLKPIN</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMARX</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRNW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCCLK</td>
</tr>
<tr>
<td>REFCCLK2</td>
</tr>
<tr>
<td>REFCCLKBSEL</td>
</tr>
<tr>
<td>REFCCLKSEL</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B108BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B108B(0:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(0:0)</td>
</tr>
<tr>
<td>TXCHARDISPVAL(0:0)</td>
</tr>
<tr>
<td>TXCHARISK(0:0)</td>
</tr>
<tr>
<td>TXDATA(7:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC8B108BUSE</td>
</tr>
<tr>
<td>TXGARBOX64B66BUSE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPolarity</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>GT10_AURORA_2</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLKBSSEL</td>
</tr>
<tr>
<td>REFCLKSEL</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B6E6USE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B6E6USE</td>
</tr>
<tr>
<td>RXDEC8B10USE</td>
</tr>
<tr>
<td>RXDESCRAM64B6E6USE</td>
</tr>
<tr>
<td>RXIGNOREBTIF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPVAL(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B6E6USE</td>
</tr>
<tr>
<td>TXENC8B10USE</td>
</tr>
<tr>
<td>TXGEARBOX64B6E6USE</td>
</tr>
<tr>
<td>TXHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B6E6USE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>GT10_AURORA_4</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>BREFCLKNIN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMAXLOCKSSEL(1:0)</td>
</tr>
<tr>
<td>RXCLK</td>
</tr>
<tr>
<td>RXCLK2</td>
</tr>
<tr>
<td>RXCKSEL</td>
</tr>
<tr>
<td>RXCSYNC64B66BUSE</td>
</tr>
<tr>
<td>RXCMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBUF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
</tr>
<tr>
<td>TXCHARISDISP(3:0)</td>
</tr>
<tr>
<td>TXCHARISDISPVAL(3:0)</td>
</tr>
<tr>
<td>TXCHARIS(3:0)</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
</tbody>
</table>

---

Usage
This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Template

GT10_AURORA_1 VHDL Code

-- Component Attribute specification for GT10_AURORA_1
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT10_AURORA_1
-- synopsys translate_off
generic (  
   ALIGN_COMMA_WORD : integer := 1;
   CHAN_BOND_LIMIT  : integer := 16;
   CHAN_BOND_MODE   : string := "OFF";
   CHAN_BOND_ONE_SHOT : boolean := FALSE;
   CHAN_BOND_SEQ_1_1 : bit_vector := "001101111100";
   CHAN_BOND_SEQ_1_2 : bit_vector := "000000000000";
   CHAN_BOND_SEQ_1_3 : bit_vector := "0001001010";
   CHAN_BOND_SEQ_1_4 : bit_vector := "0001001010";
   CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
   CHAN_BOND_SEQ_2_1 : bit_vector := "001101111100";
   CHAN_BOND_SEQ_2_2 : bit_vector := "000000000000";
   CHAN_BOND_SEQ_2_3 : bit_vector := "0001000101";
   CHAN_BOND_SEQ_2_4 : bit_vector := "0001000101";
   CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
   CHAN_BOND_SEQ_2_USE : boolean := FALSE;
   CHAN_BOND_SEQ_LEN : integer := 2;
   CLK_COR_8B10B_DE : boolean := FALSE;
   CLK_COR_MAX_LAT : integer := 36;
   CLK_COR_MIN_LAT : integer := 28;
   CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
   CLK_COR_SEQ_1_2 : bit_vector := "00100011100";
   CLK_COR_SEQ_1_3 : bit_vector := "0000000000";
   CLK_COR_SEQ_1_4 : bit_vector := "0000000000";
   CLK_COR_SEQ_1_MASK : bit_vector := "0000";
   CLK_COR_SEQ_2_1 : bit_vector := "0000000000";
   CLK_COR_SEQ_2_2 : bit_vector := "0000000000";
   CLK_COR_SEQ_2_3 : bit_vector := "0000000000";
   CLK_COR_SEQ_2_4 : bit_vector := "0000000000";
   CLK_COR_SEQ_2_MASK : bit_vector := "0000";
   CLK_COR_SEQ_2_USE : boolean := FALSE;
   CLK_COR_SEQ_DROP : boolean := FALSE;
   CLK_COR_SEQ_LEN : integer := 2;
   CLK_CORRECT_USE : boolean := TRUE;
   COMMA_10B_MASK : bit_vector := "0001111111";
   DEC_MCOMMA_DETECT : boolean := TRUE;
   DEC_PCOMMA_DETECT : boolean := TRUE;
   DEC_VALID_COMMA_ONLY : boolean := TRUE;
   MCOMMA_10B_VALUE : bit_vector := "1010000011";
   MCOMMA_DETECT : boolean := TRUE;
   PCOMMA_10B_VALUE : bit_vector := "0101111100";
   PCOMMA_DETECT : boolean := TRUE;
   PMA_PWR_CNTRL : bit_vector := "11111111";
END GT10_AURORA_1;
PMA_SPEED : string := "25_10";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (
  BREFCLKNOUT : out std_ulogic;
  BREFCLKPOUT : out std_ulogic;
  CHBONDDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(4 downto 0);
  PMARXLOCK : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(0 downto 0);
  RXCHARISK : out std_logic_vector(0 downto 0);
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_ulogic;
  RXDATA : out std_logic_vector(7 downto 0);
  RXDISPERR : out std_logic_vector(0 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(0 downto 0);
  RXREALIGN : out std_ulogic;
  RXRECLK : out std_ulogic;
  RXRUNDISP : out std_logic_vector(0 downto 0);
  TXBUFERR : out std_logic_vector(0 downto 0);
  TXKERR : out std_logic_vector(0 downto 0);
  TXN : out std_ulogic;
  TXOUTCLK : out std_ulogic;
  TXP : out std_ulogic;
  TXRUNDISP : out std_logic_vector(0 downto 0);
  BREFCLKNIN : in std_ulogic;
  BREFCLKPIN : in std_ulogic;
  CHBONDI : in std_logic_vector(4 downto 0);
  ENCHANSYNC : in std_ulogic;
  ENMCOMMAALIGN : in std_ulogic;
  ENPCOMMAALIGN : in std_ulogic;
  LOOPBACK : in std_logic_vector(1 downto 0);
  PMAINIT : in std_ulogic;
  PMAREGADDR : in std_logic_vector(5 downto 0);
  PMAREGDATAIN : in std_logic_vector(7 downto 0);
  PMAREGRW : in std_ulogic;
  PMAREGSTROBE : in std_ulogic;
  PMARXLOCKSEL : in std_logic_vector(1 downto 0);
  POWERDOWN : in std_ulogic;
  REFCLK : in std_ulogic;
  REFCLK2 : in std_ulogic;
  REFCLKBSSEL : in std_ulogic;
  REFCLKSEL : in std_ulogic;
  RXBLOCKSYNC64B66BUSE : in std_ulogic;
  RXCOMMADETUSE : in std_ulogic;
  RXDATAWIDTH : in std_logic_vector(1 downto 0);
  RXDEC64B66BUSE : in std_ulogic;
  RXDEC8B10BUSE : in std_ulogic;
  RXDESCRAM64B66BUSE : in std_ulogic;
  RXIGNOREBTF : in std_ulogic;
  RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_logic;
RXP : in std_logic;
RXPOLARITY : in std_logic;
RXRESET : in std_logic;
RXSLIDE : in std_logic;
RXUSRCLK : in std_logic;
RXUSRCLK2 : in std_logic;
TXBYPASS8B10B : in std_logic_vector(0 downto 0);
TXCHARDISPMODE : in std_logic_vector(0 downto 0);
TXCHARDISPVAL : in std_logic_vector(0 downto 0);
TXCHARISK : in std_logic_vector(0 downto 0);
TXDATA : in std_logic_vector(7 downto 0);
TXDATWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_logic;
TXENC8B10BUSE : in std_logic;
TXGEARBOX64B66BUSE : in std_logic;
TXINHIBIT : in std_logic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_logic;
TXRESET : in std_logic;
TXSCRAM64B66BUSE : in std_logic;
TXUSRCLK : in std_logic;
TXUSRCLK2 : in std_logic);

end component;

-- Component Attribute specification for GT10_AURORA_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_AURORA_1 should be placed
-- in architecture after the begin keyword

GT10_AURORA_1_INSTANCE_NAME : GT10_AURORA_1
-- synopsys translate_off
generic map (
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on
port map (
  BREFCLUD
  => user_BREFCLUD,
  BREFCLKOUT => user_BREFCLKOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDDONE => user_CHBONDDONE,
  CHBOND => user_CHBOND,
  PMAXLOCK => user_PMAXLOCK,
  RXBUFFSTATUS => user_RXBUFFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECELCCLK => user_RXRECELCCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFFERR => user_TXBUFFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXOUTCLK => user_TXOUTCLK,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLUD => user_BREFCLUD,
  BREFCLKPIN => user_BREFCLKPIN,
  CHBOND => user_CHBOND,
  ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMARXLOCKSEL => user_PMARXLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXCDATAWIDTH => user_RXCDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSERCLK => user_RXUSERCLK,
RXUSERCLK2 => user_RXUSERCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSERCLK => user_TXUSERCLK,
TXUSERCLK2 => user_TXUSERCLK2);

GT10_AURORA_2 Code
-- Component Attribute specification for GT10_AURORA_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_AURORA_2
-- synopsys translate_off
generic (ALIGN_COMMA_WORD : integer := 1;
CHAN_BOND_LIMIT : integer := 16;
CHAN_BOND_MODE : string := "OFF";
CHAN_BOND_ONE_SHOT : boolean := FALSE;
CHAN_BOND_SEQ_1_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_3 : bit_vector := "00010010100";
CHAN_BOND_SEQ_1_4 : bit_vector := "00010010100";
CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
CHAN_BOND_SEQ_2_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_3 : bit_vector := "00010001010";
CHAN_BOND_SEQ_2_4 : bit_vector := "00010001010";
CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
CHAN_BOND_SEQ_2_USE : boolean := FALSE;
CHAN_BOND_SEQ_LEN : integer := 2;
CLK_COR_BB10B_DE : boolean := FALSE;
CLK_COR_MAX_LAT : integer := 36;
CLK_COR_MIN_LAT : integer := 28;
CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
CLK_COR_SEQ_1_2 : bit_vector := "00100011100";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
CLK_COR_SEQ_1_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 2;
COMMA_10B_MASK : bit_vector := "0001111111";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "1010000011";
MCOMMA_DETECT : boolean := TRUE;
PCHAR_10B_VALUE : bit_vector := "0101111100";
PCHAR_10B_VALUE : bit_vector := "0101111110";
PCHAR_DETECT : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "11111111";
PMA_SPEED : string := "25_20";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port {
  BREFCLKNOUT : out std_ulogic;
  BREFCLKPOUT : out std_ulogic;
  CHBONDDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(4 downto 0);
  PMARXLOCK : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(1 downto 0);
  RXCHARISK : out std_logic_vector(1 downto 0);
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXDATA : out std_logic_vector(15 downto 0);
RXDISPERR : out std_logic_vector(1 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(1 downto 0);
RXREALIGN : out std_ulogic;
RXRECCCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(1 downto 0);
TXBUFERR : out std_ulogic;
TXKERR : out std_logic_vector(1 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(1 downto 0);
BREFCLKNIN : in std_ulogic;
BREFCLKPIN : in std_ulogic;
CHBONDI : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAINIT : in std_ulogic;
PMAREGADDR : in std_logic_vector(5 downto 0);
PMAREGDATAIN : in std_logic_vector(7 downto 0);
PMAREGRW : in std_ulogic;
PMAREGSTROBE : in std_ulogic;
PMARXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKBSSEL : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXBLOCKSYNC64B66BUSE : in std_ulogic;
RXCOMMADETUSE : in std_ulogic;
RXDATAWIDTH : in std_logic_vector(1 downto 0);
RXDEC64B66BUSE : in std_ulogic;
RXDEC8B10BUSE : in std_ulogic;
RXDESCRAM64B66BUSE : in std_ulogic;
RXIGNOREBTF : in std_ulogic;
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXSLIDE : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(1 downto 0);
TXCHARDISPMODE : in std_logic_vector(1 downto 0);
TXCHARDISPVAL : in std_logic_vector(1 downto 0);
TXCHARISK : in std_logic_vector(1 downto 0);
TXDATA : in std_logic_vector(15 downto 0);
TXDATAWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_ulogic;
TXENC8B10BUSE : in std_ulogic;
TXGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
end component;

-- Component Attribute specification for GT10_AURORA_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_AURORA_2 should be placed
-- in architecture after the begin keyword

GT10_AURORA_2_INSTANCE_NAME : GT10_AURORA_2
-- synopsys translate_off
generic map (   ALIGN_COMMA_WORD          => integer_value,
                 CHAN_BOND_LIMIT           => integer_value,
                 CHAN_BOND_MODE            => string_value,
                 CHAN_BOND_ONE_SHOT        => boolean_value,
                 CHAN_BOND_SEQ_1_1         => bit_value,
                 CHAN_BOND_SEQ_1_2         => bit_value,
                 CHAN_BOND_SEQ_1_3         => bit_value,
                 CHAN_BOND_SEQ_1_4         => bit_value,
                 CHAN_BOND_SEQ_1_MASK      => bit_value,
                 CHAN_BOND_SEQ_2_1         => bit_value,
                 CHAN_BOND_SEQ_2_2         => bit_value,
                 CHAN_BOND_SEQ_2_3         => bit_value,
                 CHAN_BOND_SEQ_2_4         => bit_value,
                 CHAN_BOND_SEQ_2_MASK      => bit_value,
                 CHAN_BOND_SEQ_2_USE       => boolean_value,
                 CHAN_BOND_SEQ_LEN         => integer_value,
                 CLK_COR_8B10B_DE          => boolean_value,
                 CLK_COR_MAX_LAT           => integer_value,
                 CLK_COR_MIN_LAT           => integer_value,
                 CLK_COR_SEQ_1_1           => bit_value,
                 CLK_COR_SEQ_1_2           => bit_value,
                 CLK_COR_SEQ_1_3           => bit_value,
                 CLK_COR_SEQ_1_4           => bit_value,
                 CLK_COR_SEQ_1_MASK        => bit_value,
                 CLK_COR_SEQ_2_1           => bit_value,
                 CLK_COR_SEQ_2_2           => bit_value,
                 CLK_COR_SEQ_2_3           => bit_value,
                 CLK_COR_SEQ_2_4           => bit_value,
                 CLK_COR_SEQ_2_MASK        => bit_value,
                 CLK_COR_SEQ_2_USE         => boolean_value,
                 CLK_COR_SEQ_LEN           => integer_value,
                 CLK_CORRECT_USE           => boolean_value,
                 COMMA_10B_MASK            => bit_value,
                 DEC_MCOMMA_DETECT         => boolean_value,
                 DEC_PCOMMA_DETECT         => boolean_value,
                 DEC_VALID_COMMA_ONLY      => boolean_value,
                 MCOMMA_10B_VALUE          => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on
port map (
  BREFCLKNOUT => user_BREFCLKNOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDONE => user_CHBONDONE,
  CHBONDO => user_CHBONDO,
  PMARXLOCK => user_PMARXLOCK,
  RXBUFSIZE => user_RXBUFSIZE,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCCLK => user_RXRECCCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFFERR => user_TXBUFFERR,
  TXERR => user_TXERR,
  TXN => user_TXN,
  TXOUTCLK => user_TXOUTCLK,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLKNIN => user_BREFCLKNIN,
  BREFCLKPIN => user_BREFCLKPIN,
  CHBOND1 => user_CHBOND1,
  ENCHANSYNC => user_ENCHANSYNC,
  ENMCOMMAALIGN => user_ENMCOMMAALIGN,
  ENPCOMMAALIGN => user_ENPCOMMAALIGN,
  LOOPBACK => user_LOOPBACK,
  PMAINIT => user_PMAINIT,
  PMAREGADDR => user_PMAREGADDR,
  PMAREGDATAIN => user_PMAREGDATAIN,
  PMAREGRW => user_PMAREGRW,
  PMAREGSTROBE => user_PMAREGSTROBE,
  PMARXLOCKSEL => user_PMARXLOCKSEL,
  POWERDOWN => user_POWERDOWN,
  REFCLK => user_REFCLK,
  REFCLK2 => user_REFCLK2,
  REFCLKBSEL => user_REFCLKBSEL,
  REFCLKSEL => user_REFCLKSEL,
  RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
  RXCOMMADETUSE => user_RXCOMMADETUSE,
  RXDATAWIDTH => user_RXDATAWIDTH,
  RXDEC64B66BUSE => user_RXDEC64B66BUSE,
GT10_AURORA_2 VHDL Code

-- Component Attribute specification for GT10_AURORA_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_AURORA_4
-- synopsys translate_off
generic (  
  ALIGN_COMMA_WORD : integer := 1;
  CHAN_BOND_LIMIT : integer := 16;
  CHAN_BOND_MODE : string := "OFF";
  CHAN_BOND_ONE_SHOT : boolean := FALSE;
  CHAN_BOND_SEQ_1_1 : bit_vector := "00110001100";
  CHAN_BOND_SEQ_1_2 : bit_vector := "00110001100";
  CHAN_BOND_SEQ_1_3 : bit_vector := "00110001100";
  CHAN_BOND_SEQ_1_4 : bit_vector := "00110001100";
  CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
  CHAN_BOND_SEQ_2_1 : bit_vector := "0011000000000000";
  CHAN_BOND_SEQ_2_2 : bit_vector := "0011000000000000";
  CHAN_BOND_SEQ_2_3 : bit_vector := "0011000000000000";
  CHAN_BOND_SEQ_2_4 : bit_vector := "0011000000000000";
  CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
  CHAN_BOND_SEQ_2_USE : boolean := FALSE;
  CHAN_BOND_SEQ_LEN : integer := 4;
  CLK_COR_8B10B_DE : boolean := FALSE;
  CLK_COR_MAX_LAT : integer := 36;
  CLK_COR_MIN_LAT : integer := 28;
  CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
  CLK_COR_SEQ_1_2 : bit_vector := "00100011100";
  TXDATAWIDTH : integer := 16;
  TXENC64B66BUSE : user_TXENC64B66BUSE;
  TXENC8B10BUSE : user_TXENC8B10BUSE;
  TXGEARBOX64B66BUSE : user_TXGEARBOX64B66BUSE;
  TXINHIBIT : user_TXINHIBIT;
  TXINTDATAWIDTH : user_TXINTDATAWIDTH;
  TXPOLARITY : user_TXPOLARITY;
  TXRESET : user_TXRESET;
);
CLK_COR_SEQ_1_3 : bit_vector := "00100011100";
CLK_COR_SEQ_1_4 : bit_vector := "00100011100";
CLK_COR_SEQ_1_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_1 : bit_vector := "000000000000";
CLK_COR_SEQ_2_2 : bit_vector := "000000000000";
CLK_COR_SEQ_2_3 : bit_vector := "000000000000";
CLK_COR_SEQ_2_4 : bit_vector := "000000000000";
CLK_COR_SEQ_2_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_DROP : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 4;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "000111111111";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "101000000000";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "010111110000";
PCOMMA_DETECT : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "111111111111";
PMA_SPEED : string := "25_40";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (BREFCLKNOUT : out std_ulogic;
BREFCLKPOUT : out std_ulogic;
CHBOND-done : out std_ulogic;
CHBOND : out std_logic_vector(4 downto 0);
PMA_RXLOCK : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(3 downto 0);
RXCHARISK : out std_logic_vector(3 downto 0);
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXDATA : out std_logic_vector(31 downto 0);
RXDISPERR : out std_logic_vector(3 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(3 downto 0);
RXREALIGN : out std_ulogic;
RXRECCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(3 downto 0);
TXBUFERR : out std_ulogic;
TXERR : out std_logic_vector(3 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(3 downto 0);
BREFCLKIN : in std_ulogic;
BREFCLKPIN : in std_ulogic;
CHBOND : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAINIT : in std_ulogic;
PMAREGADDR : in std_logic_vector(5 downto 0);
PMAREGDATAIN : in std_logic_vector(7 downto 0);
PMAREGRW : in std_ulogic;
PMAREGSTROBE : in std_ulogic;
PMARXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKBSEL : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXBLOCKSYNC64B66BUSE : in std_ulogic;
RXCOMMADETUSE : in std_ulogic;
RXDATAWIDTH : in std_logic_vector(1 downto 0);
RXENC64B66BUSE : in std_ulogic;
RXENC8B10BUSE : in std_ulogic;
RXDESCRAM64B66BUSE : in std_ulogic;
RXIGNOREBTF : in std_ulogic;
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXSLIDE : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(3 downto 0);
TXCHARDISPMODE : in std_logic_vector(3 downto 0);
TXCHARDISVAL : in std_logic_vector(3 downto 0);
TXCHARISK : in std_logic_vector(3 downto 0);
TXDATA : in std_logic_vector(31 downto 0);
TXDATAWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_ulogic;
TXENC8B10BUSE : in std_ulogic;
TXGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXSCRAM64B66BUSE : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT10_AURORA_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_AURORA_4 should be placed
-- in architecture after the begin keyword

GT10_AURORA_4_INSTANCE_NAME : GT10_AURORA_4
-- synopsys translate_off
generic map (  
    ALIGN_COMMA_WORD          => integer_value,  
    CHAN_BOND_LIMIT           => integer_value,  
    CHAN_BOND_MODE            => string_value,  
    CHAN_BOND_ONE_SHOT        => boolean_value,  
    CHAN_BOND_SEQ_1_1         => bit_value,  
    CHAN_BOND_SEQ_1_2         => bit_value,  
    CHAN_BOND_SEQ_1_3         => bit_value,  
    CHAN_BOND_SEQ_1_4         => bit_value,  
    CHAN_BOND_SEQ_1_MASK      => bit_value,  
    CHAN_BOND_SEQ_2_1         => bit_value,  
    CHAN_BOND_SEQ_2_2         => bit_value,  
    CHAN_BOND_SEQ_2_3         => bit_value,  
    CHAN_BOND_SEQ_2_4         => bit_value,  
    CHAN_BOND_SEQ_2_MASK      => bit_value,  
    CHAN_BOND_SEQ_2_USE       => boolean_value,  
    CHAN_BOND_SEQ_LEN         => integer_value,  
    CLK_COR_8B10B_DE          => boolean_value,  
    CLK_COR_MAX_LAT           => integer_value,  
    CLK_COR_MIN_LAT           => integer_value,  
    CLK_COR_SEQ_1_1           => bit_value,  
    CLK_COR_SEQ_1_2           => bit_value,  
    CLK_COR_SEQ_1_3           => bit_value,  
    CLK_COR_SEQ_1_4           => bit_value,  
    CLK_COR_SEQ_1_MASK        => bit_value,  
    CLK_COR_SEQ_2_1           => bit_value,  
    CLK_COR_SEQ_2_2           => bit_value,  
    CLK_COR_SEQ_2_3           => bit_value,  
    CLK_COR_SEQ_2_4           => bit_value,  
    CLK_COR_SEQ_2_MASK        => bit_value,  
    CLK_COR_SEQ_2_USE         => boolean_value,  
    CLK_COR_SEQ_DROP          => boolean_value,  
    CLK_COR_SEQ_LEN           => integer_value,  
    CLK_CORRECT_USE           => boolean_value,  
    COMMA_10B_MASK            => bit_value,  
    DEC_MCOMMA_DETECT         => boolean_value,  
    DEC_PCOMMA_DETECT         => boolean_value,  
    DEC_VALID_COMMA_ONLY      => boolean_value,  
    MCOMMA_10B_VALUE          => bit_value,  
    MCOMMA_DETECT             => boolean_value,  
    PCOMMA_10B_VALUE          => bit_value,  
    PCOMMA_DETECT             => boolean_value,  
    PMA_PWR_CNTRL             => bit_value,  
    PMA_SPEED                 => string_value,  
    RX_BUFFER_USE             => boolean_value,  
    RX_LOS_INVALID_INCR       => integer_value,  
    RX_LOS_THRESHOLD          => integer_value,  
    RX_LOSS_OF_SYNC_FSM       => boolean_value,  
    TX_BUFFER_USE             => boolean_value)  
-- synopsys translate_on  
port map (  
    BREFCLKNOUT               => user_BREFCLKNOUT,  
    BREFCLKPOUT               => user_BREFCLKPOUT,  
    CHBONDDONE                => user_CHBONDDONE,  
    CHBONO                     => user_CHBONO,  
    PMARXLOCK                  => user_PMARXLOCK,  
    RXBUFSTATUS               => user_RXBUFSTATUS,
<table>
<thead>
<tr>
<th>Setting</th>
<th>User Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXCHARISCOMMA</td>
<td>user_RXCHARISCOMMA,</td>
</tr>
<tr>
<td>RXCHARISK</td>
<td>user_RXCHARISK,</td>
</tr>
<tr>
<td>RXCLKCORCNT</td>
<td>user_RXCLKCORCNT,</td>
</tr>
<tr>
<td>RXCOMMADET</td>
<td>user_RXCOMMADET,</td>
</tr>
<tr>
<td>RXDATA</td>
<td>user_RXDATA,</td>
</tr>
<tr>
<td>RXDISPERR</td>
<td>user_RXDISPERR,</td>
</tr>
<tr>
<td>RXLOSSOFSYNC</td>
<td>user_RXLOSSOFSYNC,</td>
</tr>
<tr>
<td>RXNOTINTABLE</td>
<td>user_RXNOTINTABLE,</td>
</tr>
<tr>
<td>RXREALIGN</td>
<td>user_RXREALIGN,</td>
</tr>
<tr>
<td>RXRECCCLK</td>
<td>user_RXRECCCLK,</td>
</tr>
<tr>
<td>RXRUNDISP</td>
<td>user_RXRUNDISP,</td>
</tr>
<tr>
<td>TXBUFERR</td>
<td>user_TXBUFERR,</td>
</tr>
<tr>
<td>TXKERR</td>
<td>user_TXKERR,</td>
</tr>
<tr>
<td>TXN</td>
<td>user_TXN,</td>
</tr>
<tr>
<td>TXOUTCLK</td>
<td>user_TXOUTCLK,</td>
</tr>
<tr>
<td>TXP</td>
<td>user_TXP,</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>user_TXRUNDISP,</td>
</tr>
<tr>
<td>BREFCLKNIN</td>
<td>user_BREFCLKNIN,</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td>user_BREFCLKPIN,</td>
</tr>
<tr>
<td>CHBONDI</td>
<td>user_CHBONDI,</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>user_ENCHANSYNC,</td>
</tr>
<tr>
<td>ENNCOMMAALIGN</td>
<td>user_ENNCOMMAALIGN,</td>
</tr>
<tr>
<td>ENPFCOMMAALIGN</td>
<td>user_ENPFCOMMAALIGN,</td>
</tr>
<tr>
<td>LOOPBACK</td>
<td>user_LOOPBACK,</td>
</tr>
<tr>
<td>PMAINIT</td>
<td>user_PMAINIT,</td>
</tr>
<tr>
<td>PMAREGADDR</td>
<td>user_PMAREGADDR,</td>
</tr>
<tr>
<td>PMAREGDATAIN</td>
<td>user_PMAREGDATAIN,</td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>user_PMAREGRW,</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td>user_PMAREGSTROBE,</td>
</tr>
<tr>
<td>PMARXLOCKSEL</td>
<td>user_PMARXLOCKSEL,</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>user_POWERDOWN,</td>
</tr>
<tr>
<td>REFCLK</td>
<td>user_REFCLK,</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>user_REFCLK2,</td>
</tr>
<tr>
<td>REFCLKBSEL</td>
<td>user_REFCLKBSEL,</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>user_REFCLKSEL,</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>user_RXBLOCKSYNC64B66BUSE,</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>user_RXCOMMADETUSE,</td>
</tr>
<tr>
<td>RXDATAWIDTH</td>
<td>user_RXDATAWIDTH,</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>user_RXDEC64B66BUSE,</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>user_RXDEC8B10BUSE,</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td>user_RXDESCRAM64B66BUSE,</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>user_RXIGNOREBTF,</td>
</tr>
<tr>
<td>RXINTDATAWIDTH</td>
<td>user_RXINTDATAWIDTH,</td>
</tr>
<tr>
<td>RXN</td>
<td>user_RXN,</td>
</tr>
<tr>
<td>RXP</td>
<td>user_RXP,</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>user_RXPOLARITY,</td>
</tr>
<tr>
<td>RXRESET</td>
<td>user_RXRESET,</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>user_RXSLIDE,</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>user_RXUSRCLK,</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>user_RXUSRCLK2,</td>
</tr>
<tr>
<td>TXBYPASS8B10B</td>
<td>user_TXBYPASS8B10B,</td>
</tr>
<tr>
<td>TXCHARDISPVAL</td>
<td>user_TXCHARDISPVAL,</td>
</tr>
<tr>
<td>TXCHARISK</td>
<td>user_TXCHARISK,</td>
</tr>
<tr>
<td>TxDATA</td>
<td>user_TxDATA,</td>
</tr>
<tr>
<td>TxDATAWIDTH</td>
<td>user_TxDATAWIDTH,</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>user_TXENC64B66BUSE,</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>user_TXENC8B10BUSE,</td>
</tr>
</tbody>
</table>
TXGARBOX64B66BUSE => user_TXGARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

Verilog Instantiation Templates

GT10_AURORA_1 Verilog Code

GT10_AURORA_1 GT10_AURORA_1_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .PMARXLOCK (user_PMARXLOCK),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECCLK (user_RXRECCLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLKNIN (user_BREFCLKNIN),
  .BREFCLKPIN (user_BREFCLKPIN),
  .CHBONDI (user_CHBONDI),
  .ENCHANSYNC (user_ENCHANSYNC),
  .EMCOMMAALIGN (user_EMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .PMAINIT (user_PMAINIT),
  .PMAREGADDR (user_PMAREGADDR),
  .PMAREGDATAIN (user_PMAREGDATAIN),
  .PMAREGDATAOUT (user_PMAREGDATAOUT),
  .PMAREGRW (user_PMAREGRW),
  .PMAREGSTROBE (user_PMAREGSTROBE),
  .PMARXLOCKSEL (user_PMARXLOCKSEL),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKBSEL (user_REFCLKBSEL),
  .REFCLKSEL (user_REFCLKSEL),
.RXBLO CKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBIT (user_RXIGNOREBIT),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISP (user_TXCHARDISP),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001000101;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001000101;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 2;
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
GT10_AURORA_2 Verilog Code

GT10_AURORA_2 GT10_AURORA_2_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDONE (user_CHBONDONE),
  .CHBONDO (user_CHBONDO),
  .PMARXLOCK (user_PMARXLOCK),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECECLK (user_RXRECECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLKNIN (user_BREFCLKNIN),
  .BREFCLKPIN (user_BREFCLKPIN),
  .CHBONDI (user_CHBONDI),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .PMAINIT (user_PMAINIT),
)

defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b00011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b10100000111;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "25_10";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

Libraries Guide
ISE 6.li
www.xilinx.com
1-800-255-7778
861
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGRW (user_PMAREGRW),
.PMARESTROBE (user_PMARESTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 2;
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b0001111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "25_20";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RXLOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";

GT10_AURORA_4 Verilog Code

```
GT10_AURORA_4 GT10_AURORA_4_name(  
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONO (user_CHBONO),
  .PMARXLOCK (user_PMARXLOCK),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
```

Libraries Guide  www.xilinx.com
ISE 6.li  1-800-255-7778
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLKIN (user_BREFCLKIN),
.BREFCLKFIN (user_BREFCLKFIN),
.CHBONDI (user_CHBONDI),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.PMAINIT (user_PMAINIT),
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGRW (user_PMAREGRW),
.PMAREGSTROBE (user_PMAREGSTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPV (user_TXCHARDISPV),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2);
GT10_AURORAX_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_AURORAX_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.</td>
</tr>
</tbody>
</table>

This Xilinx protocol 10-gigabit transceiver supports 4 and 8-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 4 or 8.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the RocketIO Transceiver User Guide.
### GT10_AURORAX_4

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>Clock input</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
<td>Clock output</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>Enable channel sync</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>Enable even M comma align</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>Enable even P comma align</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>Loopback control</td>
</tr>
<tr>
<td>PMAINI</td>
<td>Power management</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>PM arbitration address</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td>PM arbitration data in</td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>PM arbitration write</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td>PM arbitration strobe</td>
</tr>
<tr>
<td>PMAXLOCKSEL(1:0)</td>
<td>PM max lock sel</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>Power down</td>
</tr>
<tr>
<td>REFCLK</td>
<td>Reference clock</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>Reference clock spare</td>
</tr>
<tr>
<td>REFCLKBSEL</td>
<td>Reference clock bypass</td>
</tr>
<tr>
<td>RXBUFSTATUS(1:0)</td>
<td>RX buffer status</td>
</tr>
<tr>
<td>RXCHARISCOMMA(3:0)</td>
<td>RX character space</td>
</tr>
<tr>
<td>RXCHARISK(3:0)</td>
<td>RX character space</td>
</tr>
<tr>
<td>RXCLKCORCNT(2:0)</td>
<td>RX clock correction count</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>RX command detect</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RX data width</td>
</tr>
<tr>
<td>RXDEC64666BUSE</td>
<td>RX decoder 64/66B use</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RX decoder 8B/10B use</td>
</tr>
<tr>
<td>RXdecoded64666BUSE</td>
<td>RX decoded 64/66B use</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>RX ignore BTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RX interrupt data width</td>
</tr>
<tr>
<td>RXN</td>
<td>RX negative</td>
</tr>
<tr>
<td>RXP</td>
<td>RX positive</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RX polarity</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RX reset</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>RX slide</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>RX user clock</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>RX user clock spare</td>
</tr>
<tr>
<td>TXCHARISCOMMA(3:0)</td>
<td>TX character space</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
<td>TX character space</td>
</tr>
<tr>
<td>TXCHARISCOMMA(3:0)</td>
<td>TX character comma</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TX character disp mode</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TX character disp mode</td>
</tr>
<tr>
<td>TXCHARISCOMMA(3:0)</td>
<td>TX character comma</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
<td>TX data</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TX data width</td>
</tr>
<tr>
<td>TXENC64666BUSE</td>
<td>TX encoder 64/66B use</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>TX encoder 8B/10B use</td>
</tr>
<tr>
<td>TXGEARBOX64666BUSE</td>
<td>TX gear box 64/66B use</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXKERR(3:0)</td>
<td>TX error</td>
</tr>
<tr>
<td>TXRUNDISP(3:0)</td>
<td>TX run disp</td>
</tr>
<tr>
<td>TXOUTCLK</td>
<td>TX output clock</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>TXN</td>
<td>TX negative</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TX polarity</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TX reset</td>
</tr>
<tr>
<td>TXSCRAM64666BUSE</td>
<td>TX scrambler 64/66B use</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TX user clock spare</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TX user clock</td>
</tr>
<tr>
<td>GT10_AURORAX_8</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td></td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>PMANIT</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(7:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td></td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>RXCHARISK(7:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td></td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK2</td>
<td></td>
</tr>
<tr>
<td>REFCLOCKSEL</td>
<td>RXDATA(63:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td></td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXDISPERR(7:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td></td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXNOTINTABLE(7:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RXRUNDISP(7:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXRECLCK</td>
</tr>
<tr>
<td>RXP</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>RXRESET</td>
<td></td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>TXBUFFERR</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXERR(7:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARDISPMODE(7:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPVAL(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARISK(7:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXDATA(63:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>TXRUNDISP(7:0)</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXINHBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Template

GT10_AURORAX_4 VHDL Code

-- Component Attribute specification for GT10_AURORAX_4
-- should be placed after architecture declaration but
-- before the begin keyword

compartment GT10_AURORAX_4
-- synopsys translate_off
generic (}
    CHAN_BOND_LIMIT : integer := 16;
    CHAN_BOND_MODE : string := "OFF";
    CHAN_BOND_ONE_SHOT : boolean := FALSE;
    CHAN_BOND_SEQ_1_1 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_2 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_3 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_4 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_1 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_2 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_3 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_4 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_USE : boolean := FALSE;
    CHAN_BOND_SEQ_LEN : integer := 8;
    CHAN_BOND_64B66B_SV : boolean := FALSE;
    CLK_COR_MAX_LAT : integer := 36;
    CLK_COR_MIN_LAT : integer := 28;
    CLK_COR_SEQ_1_1 : bit_vector := "100000111110";
    CLK_COR_SEQ_1_2 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_1_3 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_1_4 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_1_MASK : bit_vector := "0000";
    CLK_COR_SEQ_2_1 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_2 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_3 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_4 : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_MASK : bit_vector := "0000";
    CLK_COR_SEQ_2_USE : boolean := TRUE;
    CLK_COR_SEQ_DROP : boolean := FALSE;
    CLK_COR_SEQ_LEN : integer := 8;
    CLK_CORRECT_USE : boolean := TRUE;
    PMA_PWR_CNTRL : bit_vector := "11111111";
    PMA_SPEED : string := "0_32";
    RX_BUFFER_USE : boolean := TRUE;
    RX_LOS_INVALID_INCR : integer := 1;
    RX_LOS_THRESHOLD : integer := 4;
    RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
    SH_CNT_MAX : integer := 64;
    SH_INVALID_CNT_MAX : integer := 16;
    TX_BUFFER_USE : boolean := TRUE);
-- synopsys translate_on

port (
    BREFCLKNOUT : out std_ulogic;
    BREFCLKPOUT : out std_ulogic;
    CHBONDDONE : out std_ulogic;
    CHBOND : out std_logic_vector(4 downto 0);
    PMARXLOCK : out std_ulogic;
    RXBUFFSTATUS : out std_logic_vector(1 downto 0);
    RXCHARISCOMMA : out std_logic_vector(3 downto 0);
    RXCHARISK : out std_logic_vector(3 downto 0);
    RXCLKACORCNT : out std_logic_vector(2 downto 0);
    RXCOMMADET : out std_ulogic;
    RXDATA : out std_logic_vector(31 downto 0);
    RXDISPERR : out std_logic_vector(3 downto 0);
    RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
    RXNOTINTABLE : out std_logic_vector(3 downto 0);
    RXREALIGN : out std_ulogic;
    RXRECLK : out std_ulogic;
    RXRUNDISP : out std_logic_vector(3 downto 0);
    TXBUFERR : out std_logic;
    TXKERR : out std_logic_vector(3 downto 0);
    TXN : out std_logic;
    TXOUTCLK : out std_ulogic;
    TXP : out std_ulogic;
    TXRUNDISP : out std_logic_vector(3 downto 0);
    BREFCLKNIN : in std_ulogic;
    BREFCLKPIN : in std_ulogic;
    CHBONDI : in std_logic_vector(4 downto 0);
    ENCHANSYNC : in std_ulogic;
    EMMCOMMAALIGN : in std_logic;
    ENPCOMMAALIGN : in std_logic;
    LOOPBACK : in std_logic_vector(1 downto 0);
    PMAINIT : in std_ulogic;
    PMAREGADDR : in std_logic_vector(5 downto 0);
    PMAREGDATAIN : in std_logic_vector(7 downto 0);
    PMAREGRW : in std_ulogic;
    PMAREGSTROBE : in std_ulogic;
    PMARXLOCKSEL : in std_logic_vector(1 downto 0);
    POWERDOWN : in std_ulogic;
    REFCLK : in std_ulogic;
    REFCLOCK2 : in std_ulogic;
    REFCLOCKBSEL : in std_ulogic;
    REFCLOCKSEL : in std_ulogic;
    RXBLOCKSYNC64B66BUSE : in std_ulogic;
    RXCOMMADETUSE : in std_ulogic;
    RXDATAWIDTH : in std_logic_vector(1 downto 0);
    RXDECB66BUSE : in std_ulogic;
    RXDECB10BUSE : in std_ulogic;
    RXDESCRAMB66BUSE : in std_ulogic;
    RXIGNOREBTF : in std_ulogic;
    RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
    RXXN : in std_ulogic;
    RXP : in std_ulogic;
    RXPOLARITY : in std_ulogic;
    RXRESET : in std_ulogic;
    RXSLIDE : in std_ulogic;
    RXUSRCLK : in std_ulogic;
    RXUSRCLK2 : in std_ulogic;
end component;

-- Component Attribute specification for GT10_AURORAX_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_AURORAX_4 should be placed
-- in architecture after the begin keyword

GT10_AURORAX_4_INSTANCE_NAME : GT10_AURORAX_4
-- synopsys translate_off

generic map (   
CHAN_BOND_LIMIT => integer_value, 
CHAN_BOND_MODE => string_value, 
CHAN_BOND_ONE_SHOT => boolean_value, 
CHAN_BOND_SEQ_1_1 => bit_value, 
CHAN_BOND_SEQ_1_2 => bit_value, 
CHAN_BOND_SEQ_1_3 => bit_value, 
CHAN_BOND_SEQ_1_4 => bit_value, 
CHAN_BOND_SEQ_1_MASK => bit_value, 
CHAN_BOND_SEQ_2_1 => bit_value, 
CHAN_BOND_SEQ_2_2 => bit_value, 
CHAN_BOND_SEQ_2_3 => bit_value, 
CHAN_BOND_SEQ_2_4 => bit_value, 
CHAN_BOND_SEQ_2_MASK => bit_value, 
CHAN_BOND_SEQ_2_USE => boolean_value, 
CHAN_BOND_SEQ_LEN => integer_value, 
CHAN_BOND_64B66B_SV => boolean_value, 
CLK_COR_MAX_LAT => integer_value, 
CLK_COR_MIN_LAT => integer_value, 
CLK_COR_SEQ_1_1 => bit_value, 
CLK_COR_SEQ_1_2 => bit_value, 
CLK_COR_SEQ_1_3 => bit_value, 
CLK_COR_SEQ_1_4 => bit_value, 
CLK_COR_SEQ_1_MASK => bit_value, 
CLK_COR_SEQ_2_1 => bit_value, 
CLK_COR_SEQ_2_2 => bit_value, 
CLK_COR_SEQ_2_3 => bit_value, 
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SH_CNT_MAX => integer_value,
SH_INVALID_CNT_MAX => integer_value,
TX_BUFFER_USE => boolean_value

-- synopsys translate_on
port map (
  BREFCLKNOUT => user_BREFCLKNOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDDONE => user_CHBONDDONE,
  CHBONO => user_CHBONO,
  PMARXLOCK => user_PMARXLOCK,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCLKCORCORM => user_RXCLKCORCORM,
  RXCOMMADET => user_RXCOMMADET,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCLK => user_RXRECCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFERR => user_TXBUFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXOUTCLK => user_TXOUTCLK,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLKNIN => user_BREFCLKNIN,
  BREFCLKPIN => user_BREFCLKPIN,
  CHBOND1 => user_CHBOND1,
  ENCHANSYNC => user_ENCHANSYNC,
  ENMCOMMAALIGN => user_ENMCOMMAALIGN,
  ENPCOMMAALIGN => user_ENPCOMMAALIGN,
  LOOPBACK => user_LOOPBACK,
  PMAINIIT => user_PMAINIIT,
  PMAREGADDR => user_PMAREGADDR,
  PMAREGDATAIN => user_PMAREGDATAIN,
  PMAREGROW => user_PMAREGROW,
  PMAREGSTROBE => user_PMAREGSTROBE,
  PMARXLOCKSEL => user_PMARXLOCKSEL,
  POWERDOWN => user_POWERDOWN,
  REFCCLK => user_REFCCLK,
  REFCCLK2 => user_REFCCLK2,
  REFCCLKBSEL => user_REFCCLKBSEL,
  REFCCLKSEL => user_REFCCLKSEL,
GT10_AURORAX_8 Code

-- Component Attribute specification for GT10_AURORAX_8
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT10_AURORAX_8
  -- synopsys translate_off
gENERIC (
    CHAN_BOND_LIMIT : integer := 16;
    CHAN_BOND_MODE : string := "OFF";
    CHAN_BOND_ONE_SHOT : boolean := FALSE;
    CHAN_BOND_SEQ_1_1 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_1_2 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_1_3 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_1_4 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_1 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_2_2 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_2_3 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_2_4 : bit_vector := "000000000000";
    CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_USE : boolean := FALSE;
    CHAN_BOND_SEQ_LEN : integer := 8;
    CHAN_BOND_64B66B_SV : boolean := FALSE;
  );
CLK_COR_MAX_LAT : integer := 36;
CLK_COR_MIN_LAT : integer := 28;
CLK_COR_SEQ_1_1 : bit_vector := "10000011110";
CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
CLK_COR_SEQ_1_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_USE : boolean := TRUE;
CLK_COR_SEQ_DROP : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 8;
CLK_CORRECT_USE : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "11111111";
PMA_SPEED : string := "0_64";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SH_CNT_MAX : integer := 64;
SH_INVALID_CNT_MAX : integer := 16;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (}
BREFCLKNOUT : out std_ulogic;
BREFCLKPOUT : out std_ulogic;
CHBONDONE : out std_ulogic;
CHBONDO : out std_logic_vector(4 downto 0);
PMARXLOCK : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(7 downto 0);
RXCHARISK : out std_logic_vector(7 downto 0);
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_logic;
RXDATA : out std_logic_vector(63 downto 0);
RXDISPERR : out std_logic_vector(7 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(7 downto 0);
RXREALIGN : out std_ugolic;
RXRECLK : out std_ugolic;
RXRUNDISP : out std_logic_vector(7 downto 0);
TXBUFFERR : out std_ugolic;
TXKERR : out std_logic_vector(7 downto 0);
TXN : out std_ugolic;
TXOUTCLK : out std_ugolic;
TXP : out std_ugolic;
TXRUNDISP : out std_logic_vector(7 downto 0);
BREFCLKNIN : in std_ugolic;
BREFCLKPIN : in std_ugolic;
CHBONDI : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ugolic;
ENKCOMMAALIGN : in std_ugolic;
ENPCOMMAALIGN : in std_ugolic;
LOOPBACK : in std_logic_vector(1 downto 0);
-- Component Attribute specification for GT10_AURORAX_8
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_AURORAX_8 should be placed
-- in architecture after the begin keyword

GT10_AURORAX_8_INSTANCE_NAME : GT10_AURORAX_8
-- synopsys translate_off
generic map (    
    CHAN_BOND_LIMIT => integer_value, 
);
CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1  => bit_value,
CHAN_BOND_SEQ_1_2  => bit_value,
CHAN_BOND_SEQ_1_3  => bit_value,
CHAN_BOND_SEQ_1_4  => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1  => bit_value,
CHAN_BOND_SEQ_2_2  => bit_value,
CHAN_BOND_SEQ_2_3  => bit_value,
CHAN_BOND_SEQ_2_4  => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN  => integer_value,
CHAN_BOND_64B66B_SV => boolean_value,
CLK_COR_MAX_LAT    => integer_value,
CLK_COR_MIN_LAT    => integer_value,
CLK_COR_SEQ_1_1    => bit_value,
CLK_COR_SEQ_1_2    => bit_value,
CLK_COR_SEQ_1_3    => bit_value,
CLK_COR_SEQ_1_4    => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1    => bit_value,
CLK_COR_SEQ_2_2    => bit_value,
CLK_COR_SEQ_2_3    => bit_value,
CLK_COR_SEQ_2_4    => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE  => boolean_value,
CLK_COR_SEQ_DROP   => boolean_value,
CLK_COR_SEQ_LEN    => integer_value,
CLK_CORRECT_USE    => boolean_value,
PMA_PWR_CNTRL      => bit_value,
PMA_SPEED          => string_value,
RX_BUFFER_USE      => boolean_value,
RX_LOS_INVALID_INCR=> integer_value,
RX_LOS_THRESHOLD   => integer_value,
RX_LOSS_OF_SYNC_FSM=> boolean_value,
SH_CNT_MAX         => integer_value,
SH_INVALID_CNT_MAX=> integer_value,
TX_BUFFER_USE      => boolean_value)

-- synopsys translate_on
port map (  
  BREFCLKNOUT  => user_BREFCLKNOUT,
  BREFCLKPOUT  => user_BREFCLKPOUT,
  CHBONDDONE   => user_CHBONDDONE,
  CHBONO       => user_CHBONO,
  PMARXLOCK    => user_PMARXLOCK,
  RXBUFSTATUS  => user_RXBUFSTATUS,
  RXCHARISCOMMA=> user_RXCHARISCOMMA,
  RXCHARISK    => user_RXCHARISK,
  RXCLKCORCNT  => user_RXCLKCORCNT,
  RXCOMMADET   => user_RXCOMMADET,
  RXDATA       => user_RXDATA,
  RXDISPERR    => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN    => user_RXREALIGN,
Verilog Instantiation Templates

GT10_AURORAX_4 Verilog Code

GT10_AURORAX_4 GT10_AURORAX_4_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .PMARXLOCK (user_PMARXLOCK),
  .RXXBUFSTATUS (user_RXXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECECLK (user_RXRECECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLKNIN (user_BREFCLKNIN),
  .BREFCLKPIN (user_BREFCLKPIN),
  .CHBONDI (user_CHBONDI),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .PMAINIT (user_PMAINIT),
  .PMAREGADDR (user_PMAREGADDR),
  .PMAREGDATAIN (user_PMAREGDATAIN),
  .PMAREGRW (user_PMAREGRW),
  .PMAREGSTROBE (user_PMAREGSTROBE),
  .PMARXLOCKSEL (user_PMARXLOCKSEL),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKBSEL (user_REFCLKBSEL),
  .REFCLKSEL (user_REFCLKSEL),
  .RXBLOCSYNC64B66BUSE (user_RXBLOCSYNC64B66BUSE),
  .RXCOMMADETUSE (user_RXCOMMADETUSE),
  .RXDATAWIDTH (user_RXDATAWIDTH),
  .RXDEC64B66BUSE (user_RXDEC64B66BUSE),
  .RXDEC8B10BUSE (user_RXDEC8B10BUSE),
  .RXESCARAM64B66BUSE (user_RXESCARAM64B66BUSE),
  .RXIGNOREBTF (user_RXIGNOREBTF),
  .RXINTDATAWIDTH (user_RXINTDATAWIDTH),
  .RXN (user_RXN),
  .RXP (user_RXP),
)
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATWIDTH (user_TXDATWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATWIDTH (user_TXINTDATWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 8;
defparam user_instance_name.CHAN_BOND_64B66B_SV = "TRUE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "TRUE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 16;
defparam user_instance_name.CLK_COR_SEQ DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ DROP = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "0_32";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SH_CNT_MAX = 64;
defparam user_instance_name.SH_INVALID_CNT_MAX = 16;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

GT10_AURORAX_8 Verilog Code

GT10_AURORAX_8 GT10_AURORAX_8_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONO (user_CHBONO),
  .PMARKLOCK (user_PMARKLOCK),
  .RXXBUFSTATUS (user_RXXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXdISPPERR (user_RXdISPPERR),
  .RXLOSSOFFSYNC (user_RXLOSSOFFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLKNIN (user_BREFCLKNIN),
  .BREFCLKPIN (user_BREFCLKPIN),
  .CHBONDI (user_CHBONDI),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .PMAINIT (user_PMAINIT),
  .PMAREGADDR (user_PMAREGADDR),
  .PMAREGDATAIN (user_PMAREGDATAIN),
  .PMAREGRW (user_PMAREGRW),
  .PMAREGSTROBE (user_PMAREGSTROBE),
  .PMARXLOCKSEL (user_PMARXLOCKSEL),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user REFERCLK),
  .REFCLK2 (user REFERCLK2),
  .REFCLKBSEL (user REFERCLKBSEL),
  .REFCLKSEL (user REFERCLKSEL),
  .RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
  .RXCOMMADETEUSE (user_RXCOMMADETEUSE),
  .RXDATAWIDTH (user_RXDATAWIDTH),
  .RXDEC64B66BUSE (user_RXDEC64B66BUSE),
  .RXDEC8B10BUSE (user_RXDEC8B10BUSE),
  .RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
  .RXIGNOREBTF (user_RXIGNOREBTF),
  .RXIGNOREBTF (user_RXIGNOREBTF),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP))
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYTEPASS8B10B (user_TXBYTEPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_LEN = 8;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_2_LEN = 8;
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b10000000000;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_1_USE = "TRUE";
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "TRUE";
defparam user_instance_name.CLK_COR_SEQ_2_LEN = 8;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "0_64";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SH_CNT_MAX = 64;
defparam user_instance_name.SH_INVALID_CNT_MAX = 16;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

Commonly Used Constraints

None
10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_CUSTOM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This 10-gigabit transceiver is fully customizable. You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figure lists the input and output ports for all values of $n$. For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>GT10_CUSTOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATA(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMARESTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFLK2</td>
</tr>
<tr>
<td>REFLKBSSEL</td>
</tr>
<tr>
<td>REFLKSEL</td>
</tr>
<tr>
<td>RXBLKSYNC64B86BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS88B10B(7:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(7:0)</td>
</tr>
<tr>
<td>TXCHARDISPMOD(7:0)</td>
</tr>
<tr>
<td>TXCHARISK(7:0)</td>
</tr>
<tr>
<td>TXDATA(63:0)</td>
</tr>
<tr>
<td>TXDATWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Template

GT10_CUSTOM VHDL Code

-- Component Attribute specification for GT10_CUSTOM
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_CUSTOM
-- synopsys translate_off
generic ()
        ALIGN_COMMA_WORD : integer := 1;
        CHAN_BOND_LIMIT : integer := 16;
        CHAN_BOND_MODE : string := "OFF";
        CHAN_BOND_ONE_SHOT : boolean := FALSE;
        CHAN_BOND_SEQ_1_1 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_1_2 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_1_3 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_1_4 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_1_MASK : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_2_1 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_2_2 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_2_3 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_2_4 : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_2_MASK : bit_vector := "0000000000000000";
        CHAN_BOND_SEQ_2_USE : boolean := FALSE;
        CHAN_BOND_SEQ_LEN : integer := 1;
        CHAN_BOND_64B66B_SV : boolean := FALSE;
        CLK_COR_8B10B_DE : boolean := FALSE;
        CLK_COR_MAX_LAT : integer := 36;
        CLK_COR_MIN_LAT : integer := 28;
        CLK_COR_SEQ_1_1 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_1_2 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_1_3 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_1_4 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_1_MASK : bit_vector := "0000000000000000";
        CLK_COR_SEQ_2_1 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_2_2 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_2_3 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_2_4 : bit_vector := "0000000000000000";
        CLK_COR_SEQ_2_MASK : bit_vector := "0000000000000000";
        CLK_COR_SEQ_2_USE : boolean := FALSE;
        CLK_COR_SEQ_DROP : boolean := FALSE;
        CLK_COR_SEQ_LEN : integer := 1;
        CLK_CORRECT_USE : boolean := TRUE;
        COMMA_10B_MASK : bit_vector := "0001111111111111";
        DEC_MCOMMA_DETECT : boolean := TRUE;
        DEC_PCOMMA_DETECT : boolean := TRUE;
        DEC_VALID_COMMA_ONLY : boolean := TRUE;
        MCOMMA_10B_VALUE : bit_vector := "10100000011";
        MCOMMA_DETECT : boolean := TRUE;
        PCOMMA_10B_VALUE : bit_vector := "0101111100";
        PCOMMA_DETECT : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "11111111";
PMA_SPEED : string := "0_32";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SH_CNT_MAX : integer := 64;
SH_INVALID_CNT_MAX : integer := 16;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (  
  BREFCLKNOUT : out std_ulogic;
  BREFCLKPOUT : out std_ulogic;
  CHBONDDONE : out std_logic;
  CHBONDO : out std_logic_vector(4 downto 0);
  PMARXLOCK : out std_ulogic;
  RXBUFSIZE : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(7 downto 0);
  RXCHARISK : out std_logic_vector(7 downto 0);
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_logic;
  RXCOMMADETUSE : out std_logic;
  RXDATA : out std_logic_vector(63 downto 0);
  RXDISPERR : out std_logic_vector(7 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(7 downto 0);
  RXREALIGN : out std_logic;
  RXRECCLK : out std_logic;
  RXXRUN : out std_logic_vector(7 downto 0);
  TXBUFERR : out std_logic;
  TXERR : out std_logic_vector(7 downto 0);
  TXN : out std_logic;
  TXOUTCLK : out std_logic;
  TXP : out std_logic;
  TXRUN : out std_logic_vector(7 downto 0);
  BREFCLKNIN : in std_logic;
  BREFCLKPIN : in std_logic;
  CHBOND : in std_logic_vector(4 downto 0);
  ENCHAN : in std_logic;
  ENMCOMMAALIGN : in std_logic;
  ENPCOMMAALIGN : in std_logic;
  LOOPBACK : in std_logic_vector(1 downto 0);
  PMAINIT : in std_logic;
  PMAREGADDR : in std_logic_vector(5 downto 0);
  PMAREGDATAIN : in std_logic_vector(7 downto 0);
  PMAREGRW : in std_logic;
  PMAREGSTROBE : in std_logic;
  PMARXLOCKSEL : in std_logic_vector(1 downto 0);
  POWERDOWN : in std_logic;
  REFCLK : in std_logic;
  REFCLK2 : in std_logic;
  REFCLKBSEL : in std_logic;
  REFCLKSEL : in std_logic;
  RXBLOCKSYNC64B66BUSE : in std_logic;
  RXCOMMADETUSE : in std_logic;
  RXDATAWIDTH : in std_logic_vector(1 downto 0);
  RXDEC64B66BUSE : in std_logic;
  RXDEC8B10BUSE : in std_logic;
RXDESCRAM64B66BUSE : in std_logic;
RXIGNOREBTF : in std_logic;
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_logic;
RXP : in std_logic;
RXPOLARITY : in std_logic;
RXRESET : in std_logic;
RXSLIDE : in std_logic;
RXUSRCLK : in std_logic;
RXUSRCLK2 : in std_logic;
RXBYPASS8B10B : in std_logic_vector(7 downto 0);
TXCHARDISPMODE : in std_logic_vector(7 downto 0);
TXCHARDISPVAL : in std_logic_vector(7 downto 0);
TXCHARISK : in std_logic_vector(7 downto 0);
TXDATA : in std_logic_vector(63 downto 0);
TXDATAWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_logic;
TXENC8B10BUSE : in std_logic;
TXGEARBOX64B66BUSE : in std_logic;
TXINHIBIT : in std_logic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_logic;
TXRESET : in std_logic;
TXSCRAM64B66BUSE : in std_logic;
TXUSRCLK : in std_logic;
TXUSRCLK2 : in std_logic;
end component;

-- Component Attribute specification for GT10_CUSTOM
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_CUSTOM should be placed
-- in architecture after the begin keyword

GT10_CUSTOM_INSTANCE_NAME : GT10_CUSTOM
-- synopsys translate_off
generic map (ALIGN_COMMA_WORD => integer_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_64B66B_SV => boolean_value,
CLK_COR_8B10B_DE => boolean_value,
CLK_COR_MAX_LAT => integer_value,
CLK_COR_MIN_LAT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SH_CNT_MAX => integer_value,
SH_INVALID_CNT_MAX => integer_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on
port map (}
  BREFCLKNOUT => user_BREFCLKNOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDDONE => user_CHBONDDONE,
  CHBONDO => user_CHBONDO,
  PMARXLOCK => user_PMARXLOCK,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFFSYNC => user_RXLOSSOFFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCCLK => user_RXRECCCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFERR => user_TXBUFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXOUTCLK => user_TXOUTCLK,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLKIN => user_BREFCLKIN,
BREFCLKP => user_BREFCLKP,
CHBOND1 => user_CHBOND1,
ENCHANSYNC => user_ENCHANSYNC,
ENNCOMMAALIGN => user_ENNCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMARXLOCKSEL => user_PMARXLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNCE6B66BUSE => user_RXBLOCKSYNCE6B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);
Verilog Instantiation Template

GT10_CUSTOM Verilog Code

```verilog
GT10_CUSTOM GT10_CUSTOM_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .PMARXLOCK (user_PMARXLOCK),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RxDISPERR (user_RxDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLKNIN (user_BREFCLKNIN),
    .BREFCLKPIN (user_BREFCLKPIN),
    .CHBONDI (user_CHBONDI),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .PMAINIT (user_PMAINIT),
    .PMAREGADDR (user_PMAREGADDR),
    .PMAREGDATAIN (user_PMAREGDATAIN),
    .PMAREGRW (user_PMAREGRW),
    .PMAREGSTROBE (user_PMAREGSTROBE),
    .PMARXLOCKSEL (user_PMARXLOCKSEL),
    .POWERDOWN (user_POWERDOWN),
    .REFCLK (user_REFCLOCK),
    .REFCLK (user_REFCLOCK),
    .REFCLK2 (user_REFCLOCK2),
    .REFCLKBSEL (user_REFCLOCKBSEL),
    .REFCLKSEL (user_REFCLOCKSEL),
    .RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
    .RXCOMMADEUSE (user_RXCOMMADEUSE),
    .RXDATWIDTH (user_RXDATWIDTH),
    .RXDEC64B66BUSE (user_RXDEC64B66BUSE),
    .RXDEC8B10BUSE (user_RXDEC8B10BUSE),
    .RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
    .RXIGNOREBTF (user_RXIGNOREBTF),
    .RXINTDATWIDTH (user_RXINTDATWIDTH),
    .RXN (user_RXN),
    .RXP (user_RXP),
```

.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPSVAL (user_TXCHARDISPSVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATATWIDTH (user_TXDATATWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATATWIDTH (user_TXINTDATATWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 1;
defparam user_instance_name.CHAN_BOND_64B66B_SV = "FALSE";
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 1;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b00011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "0_32";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SH_CNT_MAX = 64;
defparam user_instance_name.SH_INVALID_CNT_MAX = 16;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_INFINIBAND_n</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500NV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Infiniband 10-gigabit transceiver supports 1, 2, and 4-byte data paths. The letter $n$ represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures lists the input and output ports for all values of $n$. For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>GT10_INFINIBAND_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKP</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHAN</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAIN</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCCLKBS</td>
</tr>
<tr>
<td>REFCCLKS</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66USE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66USE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66USE</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(0:0)</td>
</tr>
<tr>
<td>TXCHARISDMODE(0:0)</td>
</tr>
<tr>
<td>TXCHARISPLAYER(0:0)</td>
</tr>
<tr>
<td>TXDATA(7:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66USE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXGARBAGE64B66USE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66USE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>GT10_INFINIBAND_2</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLKBSSEL</td>
</tr>
<tr>
<td>REFCLKSEL</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
</tr>
<tr>
<td>TXCHARDISPVAL(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXGEARB6064B66BUSE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>Signal</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>REFCLKIN</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>CHBONDIO(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLOCKSEL</td>
</tr>
<tr>
<td>REFCLOCKSEL</td>
</tr>
<tr>
<td>RXCLKSYNC4866BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC4866BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
</tr>
<tr>
<td>TXCHARISPMODE(3:0)</td>
</tr>
<tr>
<td>TXCHARISPMODE(3:0)</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC4866BUSE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
</tbody>
</table>
GT10_XAUI_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_XAUI_n</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro but not for Virtex-II or Virtex-II Pro X.

This XAUI 10-gigabit transceiver supports 1, 2, and 4-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 1, 2 or 4.

You can also set attributes for the primitives. See the RocketIO Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the RocketIO Transceiver User Guide.
<table>
<thead>
<tr>
<th>GT10_XAUI_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>CHBOND(4.0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1.0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5.0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7.0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1.0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLK</td>
</tr>
<tr>
<td>REFCLK2</td>
</tr>
<tr>
<td>REFCLKXSEL</td>
</tr>
<tr>
<td>REFCLKSEL</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1.0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1.0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(0.0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(0.0)</td>
</tr>
<tr>
<td>TXCHARDISPMVAL(0.0)</td>
</tr>
<tr>
<td>TXCHARISK(0.0)</td>
</tr>
<tr>
<td>TXDATA(7.0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1.0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXGEOBOX64B66BUSE</td>
</tr>
<tr>
<td>TXINHBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1.0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>GT10_XAUI_2</td>
</tr>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMARESTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLOCK</td>
</tr>
<tr>
<td>REFCLOCK2</td>
</tr>
<tr>
<td>REFCLOCKBSEL</td>
</tr>
<tr>
<td>REFCLOCKSEL</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDEB</td>
</tr>
<tr>
<td>RXUSRCCLK</td>
</tr>
<tr>
<td>RXUSRCLOCK2</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
</tr>
<tr>
<td>TXCHARISDISPMODE(1:0)</td>
</tr>
<tr>
<td>TXCHARISDISPMODEVAL(1:0)</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
</tr>
<tr>
<td>TXENBOX64B66BUSE</td>
</tr>
<tr>
<td>TXINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXP</td>
</tr>
<tr>
<td>TXR</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCCLK</td>
</tr>
<tr>
<td>TXUSRCLOCK2</td>
</tr>
<tr>
<td>Signal</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>CHBOND(4.0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
</tr>
<tr>
<td>REFCLOCK</td>
</tr>
<tr>
<td>REFCLOCK2</td>
</tr>
<tr>
<td>REFCLKBSSEL</td>
</tr>
<tr>
<td>REFCLKSEL</td>
</tr>
<tr>
<td>RXBLOCKSYNC4866BUSE</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDATA24866BUSE</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC810BUSE</td>
</tr>
<tr>
<td>RXDECSCRAM64B66BUSE</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRESET</td>
</tr>
<tr>
<td>RXSLIDE</td>
</tr>
<tr>
<td>RXUSRCLK</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
</tr>
<tr>
<td>TXBYPASS810B(3:0)</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
</tr>
<tr>
<td>TXCHARDISPV(3:0)</td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
</tr>
<tr>
<td>TXENC810BUSE</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
</tr>
<tr>
<td>TXNINHIBIT</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT10_XAUI_1 VHDL Code

```vhdl
-- Component Attribute specification for GT10_XAUI_1
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_XAUI_1
-- synopsys translate_off
generic(
  ALIGN_COMMA_WORD : integer := 2;
  ALIGN_COMMA_MODE : string := "OFF";
  CHAN_BOND_LIMIT  : integer := 16;
  CHAN_BOND_MODE   : string := "OFF";
  CHAN_BOND_ONE_SHOT: boolean := FALSE;
  CHAN_BOND_SEQ_1_1: bit_vector := "00101111100";
  CHAN_BOND_SEQ_1_2: bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_3: bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_4: bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_MASK : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_USE : boolean := FALSE;
  CHAN_BOND_SEQ_2_MASK : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_USE : boolean := FALSE;
  CHAN_BOND_SEQ_LEN : integer := 2;
  CLK_COR_8B10B_DE : boolean := FALSE;
  CLK_COR_MAX_LAT : integer := 36;
  CLK_COR_MIN_LAT : integer := 28;
  CLK_COR_SEQ_1_1 : bit_vector := "00100111100";
  CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
  CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
  CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
  CLK_COR_SEQ_1_MASK : bit_vector := "00000000000";
  CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
  CLK_COR_SEQ_2_MASK : bit_vector := "00000000000";
  CLK_COR_SEQ_2_USE : boolean := FALSE;
  CLK_COR_SEQ_2_DROP : boolean := FALSE;
  CLK_COR_SEQ_LEN : integer := 2;
  CLK_CORRECT_USE   : boolean := TRUE;
  COMMA_10B_MASK    : bit_vector := "0001111111";
  DEC_MCOMMA_DETECT : boolean := TRUE;
  DEC_PCOMMA_DETECT : boolean := TRUE;
  DEC_VALID_COMMA_ONLY : boolean := TRUE;
  MCOMMA_10B_VALUE : bit_vector := "10100000011";
  MCOMMA_DETECT    : boolean := TRUE;
  PCOMMA_10B_VALUE : bit_vector := "0101111100";
  PCOMMA_DETECT    : boolean := TRUE;
  PMA_PWR_CNTRL    : bit_vector := "11111111";
);
PMA_SPEED : string := "25_10";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port(
  BREFCLKNOUT : out std_ulogic;
  BREFCLKPOUT : out std_ulogic;
  CHBONDDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(4 downto 0);
  PMARXLOCK : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(0 downto 0);
  RXCHARISK : out std_logic_vector(0 downto 0);
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
  RXCOMMADET : out std_ulogic;
  RXDATA : out std_logic_vector(7 downto 0);
  RXDISPERR : out std_logic_vector(0 downto 0);
  RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
  RXNOTINTABLE : out std_logic_vector(0 downto 0);
  RXREALIGN : out std_ulogic;
  RXRECCLK : out std_ulogic;
  RXRUNDISP : out std_logic_vector(0 downto 0);
  TXBUFERR : out std_logic_vector(0 downto 0);
  TXKERR : out std_logic_vector(0 downto 0);
  TXN : out std_ulogic;
  TXOUTCLK : out std_ulogic;
  TXP : out std_ulogic;
  TXRUNDISP : out std_logic_vector(0 downto 0);
  BREFCLKNIN : in std_ulogic;
  BREFCLKPIN : in std_ulogic;
  CHBONDI : in std_logic_vector(4 downto 0);
  ENCHANSYNC : in std_ulogic;
  ENMCOMMAALIGN : in std_ulogic;
  ENPCOMMAALIGN : in std_ulogic;
  LOOPBACK : in std_logic_vector(1 downto 0);
  PMAINIT : in std_ulogic;
  PMAREGADDR : in std_logic_vector(5 downto 0);
  PMAREGDATAIN : in std_logic_vector(7 downto 0);
  PMAREGRW : in std_ulogic;
  PMAREGSTROBE : in std_ulogic;
  PMAXLOCKSEL : in std_logic_vector(1 downto 0);
  POWERDOWN : in std_ulogic;
  REFCLK : in std_ulogic;
  REFCLK2 : in std_ulogic;
  REFCLKBSEL : in std_ulogic;
  REFCLKSEL : in std_ulogic;
  RXBLOCKSYNC64B66BUSE : in std_ulogic;
  RXCOMMADETUSE : in std_ulogic;
  RXDATAWIDTH : in std_logic_vector(1 downto 0);
  RXDEC64B66BUSE : in std_ulogic;
  RXDEC8B10BUSE : in std_ulogic;
  RXDESCRAM64B66BUSE : in std_ulogic;
  RXIGNOREBTF : in std_ulogic;
  RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN       : in std_logic;
RXP       : in std_logic;
RXPOLARITY: in std_logic;
RXRESET   : in std_logic;
RXSLIDE   : in std_logic;
RXUSRCLK  : in std_logic;
RXUSRCLK2 : in std_logic;
TXBIYPASS8B10B  : in std_logic_vector(0 downto 0);
TXCHARDISPMODE   : in std_logic_vector(0 downto 0);
TXCHARDISVPVAL   : in std_logic_vector(0 downto 0);
TXCHARISK       : in std_logic_vector(0 downto 0);
TXDATA         : in std_logic_vector(7 downto 0);
TXDATAWIDTH    : in std_logic_vector(1 downto 0);
TXENC64B66BUSE  : in std_logic;
TXENC8B10BUSE  : in std_logic;
TXGEARBOX64B66BUSE : in std_logic;
TXINHIBIT     : in std_logic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY    : in std_logic;
TXRESET       : in std_logic;
TXSCRAM64B66BUSE : in std_logic;
TXUSRCLK      : in std_logic;
TXUSRCLK2     : in std_logic);

end component;

-- Component Attribute specification for GT10_XAUI_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_XAUI_1 should be placed
-- in architecture after the begin keyword

GT10_XAUI_1_INSTANCE_NAME : GT10_XAUI_1
-- synopsys translate_off
generic map (   ALIGN_COMMA_WORD       => integer_value,
                 CHAN_BOND_LIMIT     => integer_value,
                 CHAN_BOND_MODE      => string_value,
                 CHAN_BOND_ONE_SHOT  => boolean_value,
                 CHAN_BOND_SEQ_1_1   => bit_value,
                 CHAN_BOND_SEQ_1_2   => bit_value,
                 CHAN_BOND_SEQ_1_3   => bit_value,
                 CHAN_BOND_SEQ_1_4   => bit_value,
                 CHAN_BOND_SEQ_1_MASK=> bit_value,
                 CHAN_BOND_SEQ_2_1   => bit_value,
                 CHAN_BOND_SEQ_2_2   => bit_value,
                 CHAN_BOND_SEQ_2_3   => bit_value,
                 CHAN_BOND_SEQ_2_4   => bit_value,
                 CHAN_BOND_SEQ_2_MASK=> bit_value,
                 CHAN_BOND_SEQ_2_USE => boolean_value,
                 CHAN_BOND_SEQ_LEN   => integer_value,
                 CLK_COR_8B10B_DE    => boolean_value,
                 CLK_COR_MAX_LAT    => integer_value,
                 CLK_COR_MIN_LAT    => integer_value,
                 CLK_COR_SEQ_1_1    => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on
port map (}
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMARKLOCKSEL => user_PMARKLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT10_XAUI_2 Code

-- Component Attribute specification for GT10_XAUI_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_XAUI_2
-- synopsys translate_off
generic (  
ALIGN_COMMA_WORD : integer := 2;
CHAN_BOND_LIMIT : integer := 16;
CHAN_BOND_MODE : string := "OFF";
CHAN_BOND_ONE_SHOT      : boolean := FALSE;
CHAN_BOND_SEQ_1_1      : bit_vector := "00101111100";
CHAN_BOND_SEQ_1_2      : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_3      : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_4      : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_MASK   : bit_vector := "0000";
CHAN_BOND_SEQ_2_1      : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_2      : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_3      : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_4      : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_MASK   : bit_vector := "0000";
CHAN_BOND_SEQ_2_USE    : boolean := FALSE;
CHAN_BOND_SEQ_2_LEN    : integer := 2;
CLK_COR_8B10B_DE       : boolean := FALSE;
CLK_COR_MAX_LAT        : integer := 36;
CLK_COR_MIN_LAT        : integer := 28;
CLK_COR_SEQ_1_1        : bit_vector := "00100011100";
CLK_COR_SEQ_1_2        : bit_vector := "00000000000";
CLK_COR_SEQ_1_3        : bit_vector := "00000000000";
CLK_COR_SEQ_1_4        : bit_vector := "00000000000";
CLK_COR_SEQ_1_MASK     : bit_vector := "0000";
CLK_COR_SEQ_2_1        : bit_vector := "00000000000";
CLK_COR_SEQ_2_2        : bit_vector := "00000000000";
CLK_COR_SEQ_2_3        : bit_vector := "00000000000";
CLK_COR_SEQ_2_4        : bit_vector := "00000000000";
CLK_COR_SEQ_2_MASK     : bit_vector := "0000";
CLK_COR_SEQ_2_USE      : boolean := FALSE;
CLK_COR_SEQ_DROP       : boolean := FALSE;
CLK_COR_SEQ_LEN        : integer := 2;
CLK_CORRECT_USE        : boolean := TRUE;
COMMA_10B_MASK         : bit_vector := "0001111111";
DEC_MCOMMA_DETECT      : boolean := TRUE;
DEC_PCOMMA_DETECT      : boolean := TRUE;
DEC_VALID_COMMA_ONLY   : boolean := TRUE;
MCOMMA_10B_VALUE       : bit_vector := "1010000011";
MCOMMA_DETECT          : boolean := TRUE;
PCOMMA_10B_VALUE       : bit_vector := "0101111100";
PCOMMA_DETECT          : boolean := TRUE;
PMA_PWR_CNTRL          : bit_vector := "11111111";
PMA_SPEED              : string := "25_20";
RX_BUFFER_USE          : boolean := TRUE;
RX_LOS_INVALID_INCR    : integer := 1;
RX_LOS_THRESHOLD       : integer := 4;
RX_LOSS_OF_SYNC_FSM    : boolean := TRUE;
TX_BUFFER_USE          : boolean := TRUE;

-- synopsys translate_on

port (  
  BREFCLKNOUT               : out std_ulogic;
  BREFCLKPOUT               : out std_ulogic;
  CHBONDDONE                : out std_logic;
  CHBONO                     : out std_logic_vector(4 downto 0);
  PMARXLOCK                   : out std_ulogic;
  RXBUFSTATUS                : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA              : out std_logic_vector(1 downto 0);
  RXCHARISK                   : out std_logic_vector(1 downto 0);
  RXCLKCORCNT                : out std_logic_vector(2 downto 0);
  RXCOMMADET                 : out std_ulogic;
RXDATA : out std_logic_vector(15 downto 0);
RXDISPERR : out std_logic_vector(1 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(1 downto 0);
RXREALIGN : out std_ulogic;
RXRECLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(1 downto 0);
TXBUFERR : out std_ulogic;
TXKERR : out std_logic_vector(1 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(1 downto 0);
BREFCLKIN : in std_ulogic;
BREFCLKP : in std_ulogic;
CHBOND : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAIN : in std_ulogic;
PMAREGADDR : in std_logic_vector(5 downto 0);
PMAREGDATAIN : in std_logic_vector(7 downto 0);
PMAREGW : in std_ulogic;
PMAREGSTROBE : in std_logic;
PMAXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_logic;
REFCLK2 : in std_logic;
REFCLKB : in std_logic;
RXBLOCKSYNC64B66BUSE : in std_logic;
RXCOMMADETUSE : in std_logic;
RXDATAWIDTH : in std_logic_vector(1 downto 0);
RXDEC64B66BUSE : in std_ulogic;
RXDEC8B10BUSE : in std_ulogic;
RXDESCRAM64B66BUSE : in std_ulogic;
RXENET : in std_logic;
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXSLIDE : in std_ulogic;
RXUSRCCLK : in std_ulogic;
RXUSRCCLK2 : in std_ulogic;
TBXHYPB8B10B : in std_logic_vector(1 downto 0);
TBXCHARDISPMODE : in std_logic_vector(1 downto 0);
TBXCHARDISPV : in std_logic_vector(1 downto 0);
TXCHARISK : in std_logic_vector(1 downto 0);
TDATA : in std_logic_vector(15 downto 0);
TDATAWIDTH : in std_logic_vector(1 downto 0);
TDEC64B66BUSE : in std_ulogic;
TDEC8B10BUSE : in std_ulogic;
TGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_ulogic;
TXRESET : in std_logic;
TXSCRAM64B66BUSE : in std_logic;
TXUSRCLK : in std_logic;
TXUSRCLK2 : in std_logic);

end component;

-- Component Attribute specification for GT10_XAUI_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_XAUI_2 should be placed
-- in architecture after the begin keyword

GT10_XAUI_2_INSTANCE_NAME : GT10_XAUI_2
-- synopsys translate_off
generic map (ALIGN_COMMA_WORD          => integer_value,
          CHAN_BOND_LIMIT           => integer_value,
          CHAN_BOND_MODE            => string_value,
          CHAN_BOND_ONE_SHOT        => boolean_value,
          CHAN_BOND_SEQ_1_1         => bit_value,
          CHAN_BOND_SEQ_1_2         => bit_value,
          CHAN_BOND_SEQ_1_3         => bit_value,
          CHAN_BOND_SEQ_1_4         => bit_value,
          CHAN_BOND_SEQ_1_MASK      => bit_value,
          CHAN_BOND_SEQ_2_1         => bit_value,
          CHAN_BOND_SEQ_2_2         => bit_value,
          CHAN_BOND_SEQ_2_3         => bit_value,
          CHAN_BOND_SEQ_2_4         => bit_value,
          CHAN_BOND_SEQ_2_MASK      => bit_value,
          CHAN_BOND_SEQ_2_USE       => boolean_value,
          CHAN_BOND_SEQ_LEN         => integer_value,
          CLK_COR_8B10B_DE          => boolean_value,
          CLK_COR_MAX_LAT           => integer_value,
          CLK_COR_MIN_LAT           => integer_value,
          CLK_COR_SEQ_1_1           => bit_value,
          CLK_COR_SEQ_1_2           => bit_value,
          CLK_COR_SEQ_1_3           => bit_value,
          CLK_COR_SEQ_1_4           => bit_value,
          CLK_COR_SEQ_1_MASK        => bit_value,
          CLK_COR_SEQ_2_1           => bit_value,
          CLK_COR_SEQ_2_2           => bit_value,
          CLK_COR_SEQ_2_3           => bit_value,
          CLK_COR_SEQ_2_4           => bit_value,
          CLK_COR_SEQ_2_MASK        => bit_value,
          CLK_COR_SEQ_2_USE         => boolean_value,
          CLK_COR_SEQ_LEN           => integer_value,
          CLK_CORRECT_USE           => boolean_value,
          COMMA_10B_MASK            => bit_value,
          DEC_MCOMMA_DETECT         => boolean_value,
          DEC_PCOMMA_DETECT         => boolean_value,
          DEC_VALID_COMMA_ONLY      => boolean_value,
          MCOMMA_10B_VALUE          => bit_value,
          MCOMMA_DETECT             => boolean_value,
PCOMMA_10B_VALUE  => bit_value,
PCOMMA_DETECT     => boolean_value,
PMA_PWR_CNTRL     => bit_value,
PMA_SPEED         => string_value,
RX_BUFFER_USE     => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD  => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
TX_BUFFER_USE     => boolean_value)

-- synopsys translate_on

port map (    
  BREFCLKNOUT     => user_BREFCLKNOUT,
  BREFCLKPOUT     => user_BREFCLKPOUT,
  CHBONDDONE      => user_CHBONDDONE,
  CHBONO          => user_CHBONO,
  PMARXLOCK       => user_PMARXLOCK,
  RXBUFSTATUS     => user_RXBUFSTATUS,
  RXCHARISCOMMA   => user_RXCHARISCOMMA,
  RXCHARISK       => user_RXCHARISK,
  RXCLKCORCNT     => user_RXCLKCORCNT,
  RXCOMMADET      => user_RXCOMMADET,
  RXDATA          => user_RXDATA,
  RXDISPERR       => user_RXDISPERR,
  RXLOSSOFSYNC    => user_RXLOSSOFSYNC,
  RXNOTINTABLE    => user_RXNOTINTABLE,
  RXREALIGN       => user_RXREALIGN,
  RXRECLK        => user_RXRECLK,
  RXRUNDISP       => user_RXRUNDISP,
  TXBUFFERR       => user_TXBUFFERR,
  TXKERR          => user_TXKERR,
  TXN             => user_TXN,
  TXOUTCLK        => user_TXOUTCLK,
  TXP             => user_TXP,
  TXRUNDISP       => user_TXRUNDISP,
  BREFCLKNIN      => user_BREFCLKNIN,
  BREFCLKPIN      => user_BREFCLKPIN,
  CHBONDI         => user_CHBONDI,
  ENCHANSYNC      => user_ENCHANSYNC,
  ENMCOMMAALIGN   => user_ENMCOMMAALIGN,
  ENPCOMMAALIGN   => user_ENPCOMMAALIGN,
  LOOPBACK        => user_LOOPBACK,
  PMAINIT         => user_PMAINIT,
  PMAREGADDR      => user_PMAREGADDR,
  PMAREGDATAIN    => user_PMAREGDATAIN,
  PMAREGRW        => user_PMAREGRW,
  PMAREGSTROBE    => user_PMAREGSTROBE,
  PMARXLOCKSEL    => user_PMARXLOCKSEL,
  POWERDOWN       => user_POWERDOWN,
  REFCLK          => user_REFCLK,
  REFCLK2         => user_REFCLK2,
  REFCLKBSEL      => user_REFCLKBSEL,
  REFCLKSEL       => user_REFCLKSEL,
  RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
  RXCOMMADETUSE   => user_RXCOMMADETUSE,
  RXDATAWIDTH     => user_RXDATAWIDTH,
  RXDEC64B66BUSE  => user_RXDEC64B66BUSE,
  RXDEC8B10BUSE   => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

-- Component Attribute specification for GT10_XAUI_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_XAUI_4
-- synopsys translate_off
generic (
  ALIGN_COMMA_WORD : integer := 2;
  CHAN_BOND_LIMIT : integer := 16;
  CHAN_BOND_MODE : string := "OFF";
  CHAN_BOND_ONE_SHOT : boolean := FALSE;
  CHAN_BOND_SEQ_1_1 : bit_vector := "00101111100";
  CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_3 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_4 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
  CHAN_BOND_SEQ_2_1 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_4 : bit_vector := "00000000000";
  CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
  CHAN_BOND_SEQ_2_USE : boolean := FALSE;
  CHAN_BOND_SEQ_LEN : integer := 2;
  CLK_COR_8B10B_DE : boolean := FALSE;
  CLK_COR_MAX_LAT : integer := 36;
  CLK_COR_MIN_LAT : integer := 28;
  CLK_COR_SEQ_1_1 : bit_vector := "00100000000011100";
  CLK_COR_SEQ_1_2 : bit_vector := "00000000000000000";
  CLK_COR_SEQ_1_3 : bit_vector := "00000000000000000";
  CLK_COR_SEQ_1_4 : bit_vector := "00000000000000000";
  CLK_COR_SEQ_1_MASK : bit_vector := "0000";
)}
CLK_COR_SEQ_2_1 : bit_vector := "000000000000";
CLK_COR_SEQ_2_2 : bit_vector := "000000000000";
CLK_COR_SEQ_2_3 : bit_vector := "000000000000";
CLK_COR_SEQ_2_4 : bit_vector := "000000000000";
CLK_COR_SEQ_2_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_DROP : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 2;
CLK_CORRECT_USE : boolean := TRUE;
COMMA_10B_MASK : bit_vector := "0011111111";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "1010000011";
MCOMMA_DETECT : boolean := TRUE;
Pcomma_10B_VALUE : bit_vector := "0101111100";
Pcomma_DETECT : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "11111111";
PMA_SPEED : string := "25_40";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (}
BREFCLKNOUT : out std_ulogic;
BREFCLKPOUT : out std_ulogic;
CHBONDONE : out std_ulogic;
CHBONDO : out std_logic_vector(4 downto 0);
PMARXLOCK : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(3 downto 0);
RXCHARISK : out std_logic_vector(3 downto 0);
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_logic;
RXDATA : out std_logic_vector(31 downto 0);
RXDISPERR : out std_logic_vector(3 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(3 downto 0);
RXREALIGN : out std_ulogic;
RXRECCCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(3 downto 0);
TXBUFFERR : out std_ulogic;
TXKERR : out std_logic_vector(3 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXF : out std_ulogic;
TXRUNDISP : out std_logic_vector(3 downto 0);
BREFCLKNIN : in std_ulogic;
BREFCLKPIN : in std_ulogic;
CHBONDI : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAINIT : in std_ulogic;
PMAREGADDR : in std_logic_vector(5 downto 0);
PMAREGDATAIN : in std_logic_vector(7 downto 0);
PMAREGRW : in std_logic;
PMAREGSTROBE : in std_logic;
PMARXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_logic;
REFCLK : in std_logic;
REFCLK2 : in std_logic;
REFCLKBSEL : in std_logic;
REFCLKSEL : in std_logic;
RXBLOCKSYNC64B66BUSE : in std_logic;
RXCOMMANDUSE : in std_logic;
RXDATALength : in std_logic_vector(1 downto 0);
RXDEC64B66BUSE : in std_logic;
RXDEC8B10BUSE : in std_logic;
RXDESCRAM64B66BUSE : in std_logic;
RXIGNOREBTF : in std_logic;
RXINTDATALength : in std_logic_vector(1 downto 0);
RXN : in std_logic;
RXP : in std_logic;
RXPOLARITY : in std_logic;
RXRESET : in std_logic;
RXSLIDE : in std_logic;
RXUSRCLK : in std_logic;
RXUSRCLK2 : in std_logic;
TXBYPASS8B10B : in std_logic_vector(3 downto 0);
TXCHARDISPMODE : in std_logic_vector(3 downto 0);
TXCHARDISPVAL : in std_logic_vector(3 downto 0);
TXCHARISK : in std_logic_vector(3 downto 0);
TXDATA : in std_logic_vector(31 downto 0);
TXDATALength : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_logic;
TXENC8B10BUSE : in std_logic;
TXGEARBOX64B66BUSE : in std_logic;
TXHABIT : in std_logic;
TXINTDATALength : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_logic;
TXRESET : in std_logic;
TXSCRAM64B66BUSE : in std_logic;
TXUSRCLK : in std_logic;
TXUSRCLK2 : in std_logic;

end component;

-- Component Attribute specification for GT10_XAU1_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_XAU1_4 should be placed
-- in architecture after the begin keyword

GT10_XAU1_4_INSTANCE_NAME : GT10_XAU1_4
-- synopsys translate_off
generic map (ALIGN_COMMA_WORD => integer_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CLK_COR_8B10B_DE => boolean_value,
CLK_COR_MAX_LAT => integer_value,
CLK_COR_MIN_LAT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
COMMA_10B_MASK => bit_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
PCOMMA_10B_VALUE => bit_value,
PCOMMA_DETECT => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on
port map (  
BREFCLKNOUT => user_BREFCLKNOUT,
BREFCLKPOUT => user_BREFCLKPOUT,
CHBONDDONE => user_CHBONDDONE,
CHBONDO => user_CHBONDO,
PMARXLOCK => user_PMARXLOCK,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCLKCORCNT => user_RXCLKCORCNT,
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Corresponding Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXCOMMADET</td>
<td>user_RXCOMMADET,</td>
</tr>
<tr>
<td>RXDATA</td>
<td>user_RXDATA,</td>
</tr>
<tr>
<td>RXDISPERR</td>
<td>user_RXDISPERR,</td>
</tr>
<tr>
<td>RXLOSSOSYSYNC</td>
<td>user_RXLOSSOSYSYNC,</td>
</tr>
<tr>
<td>RXNOTINTABLE</td>
<td>user_RXNOTINTABLE,</td>
</tr>
<tr>
<td>RXREALIGN</td>
<td>user_RXREALIGN,</td>
</tr>
<tr>
<td>RXRECLK</td>
<td>user_RXRECLK,</td>
</tr>
<tr>
<td>RXRUNDISP</td>
<td>user_RXRUNDISP,</td>
</tr>
<tr>
<td>TXBUFERR</td>
<td>user_TXBUFERR,</td>
</tr>
<tr>
<td>TXKERR</td>
<td>user_TXKERR,</td>
</tr>
<tr>
<td>TXN</td>
<td>user_TXN,</td>
</tr>
<tr>
<td>TXOUTCLK</td>
<td>user_TXOUTCLK,</td>
</tr>
<tr>
<td>TXF</td>
<td>user_TXF,</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>user_TXRUNDISP,</td>
</tr>
<tr>
<td>BREFCLKNIN</td>
<td>user_BREFCLKNIN,</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td>user_BREFCLKPIN,</td>
</tr>
<tr>
<td>CHBONDI</td>
<td>user_CHBONDI,</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td>user_ENCHANSYNC,</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>user_ENMCOMMAALIGN,</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td>user_ENPCOMMAALIGN,</td>
</tr>
<tr>
<td>LOOPBACK</td>
<td>user_LOOPBACK,</td>
</tr>
<tr>
<td>PMAINIT</td>
<td>user_PMAINIT,</td>
</tr>
<tr>
<td>PMAREGADDR</td>
<td>user_PMAREGADDR,</td>
</tr>
<tr>
<td>PMAREGDATAIN</td>
<td>user_PMAREGDATAIN,</td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>user_PMAREGRW,</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td>user_PMAREGSTROBE,</td>
</tr>
<tr>
<td>PMARXLOCKSEL</td>
<td>user_PMARXLOCKSEL,</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>user_POWERDOWN,</td>
</tr>
<tr>
<td>REFCLK</td>
<td>user_REFCLK,</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>user_REFCLK2,</td>
</tr>
<tr>
<td>REFCLOCKSEL</td>
<td>user_REFCLOCKSEL,</td>
</tr>
<tr>
<td>RXBROCKSYNC64B66BUSE</td>
<td>user_RXBROCKSYNC64B66BUSE,</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>user_RXCOMMADETUSE,</td>
</tr>
<tr>
<td>RXDATAWIDTH</td>
<td>user_RXDATAWIDTH,</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>user_RXDEC64B66BUSE,</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>user_RXDEC8B10BUSE,</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td>user_RXDESCRAM64B66BUSE,</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>user_RXIGNOREBTF,</td>
</tr>
<tr>
<td>RXINTDATAWIDTH</td>
<td>user_RXINTDATAWIDTH,</td>
</tr>
<tr>
<td>RXN</td>
<td>user_RXN,</td>
</tr>
<tr>
<td>RXP</td>
<td>user_RXP,</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>user_RXPOLARITY,</td>
</tr>
<tr>
<td>RXRESET</td>
<td>user_RXRESET,</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>user_RXSLIDE,</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>user_RXUSRCLK,</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>user_RXUSRCLK2,</td>
</tr>
<tr>
<td>TXBYPASS8B10B</td>
<td>user_TXBYPASS8B10B,</td>
</tr>
<tr>
<td>TXCHARDISPMODE</td>
<td>user_TXCHARDISPMODE,</td>
</tr>
<tr>
<td>TXCHARDISPVAL</td>
<td>user_TXCHARDISPVAL,</td>
</tr>
<tr>
<td>TXCHARISK</td>
<td>user_TXCHARISK,</td>
</tr>
<tr>
<td>TXDATA</td>
<td>user_TXDATA,</td>
</tr>
<tr>
<td>TXDATAWIDTH</td>
<td>user_TXDATAWIDTH,</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>user_TXENC64B66BUSE,</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>user_TXENC8B10BUSE,</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
<td>user_TXGEARBOX64B66BUSE,</td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td>user_TXINHIBIT,</td>
</tr>
<tr>
<td>TXINTDATAWIDTH</td>
<td>user_TXINTDATAWIDTH,</td>
</tr>
</tbody>
</table>
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

End Here

Verilog Instantiation Templates

GT10_XAUI_1 Verilog Code

GT10_XAUI_1 (GT10_XAUI_1_name,
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .PMARKLOCK (user_PMARKLOCK),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLKNIN (user_BREFCLKNIN),
    .BREFCLKPIN (user_BREFCLKPIN),
    .CHBONDI (user_CHBONDI),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .PMAINIT (user_PMAINIT),
    .PMAREGADDR (user_PMAREGADDR),
    .PMAREGDATAIN (user_PMAREGDATAIN),
    .PMAREGWR (user_PMAREGWR),
    .PMAREGSTROBE (user_PMAREGSTROBE),
    .PMARKLOCKSEL (user_PMARKLOCKSEL),
    .POWERDOWN (user_POWERDOWN),
    .REFCLK (user_REFCLK),
    .REFCLK2 (user_REFCLK2),
    .REFCLKBSEL (user_REFCLKBSEL),
    .REFCLKSEL (user_REFCLKSEL),
    .RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
    .RXCOMMADETUSE (user_RXCOMMADETUSE),
    ...
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSRAEM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 2;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00101111010;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_2_LEN = 2;
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b0001111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "25_10";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.RX_REALIGN_USE = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
defparam user_instance_name.TX_REALIGN_USE = "TRUE";
.PMAREGRW (user_PMAREGRW),
.PMAREGSTROBE (user_PMAREGSTROBE),
.PMARKLOCKSEL (user_PMARKLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARISLO (user_TXCHARISLO),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGARBOX64B66BUSE (user_TXGARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 2;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b001011111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 2;
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_DROP = "FALSE"

defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE"
defparam user_instance_name.COMMA_10B_MASK = 10'b0001111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE"
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE"
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE"
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE"
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE"
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "25_20"
defparam user_instance_name.RX_BUFFER_USE = "TRUE"
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE"
defparam user_instance_name.TX_BUFFER_USE = "TRUE"

GT10_XAUI_4 Verilog Code

GT10_XAUI_4 GT10_XAUI_4_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDONE (user_CHBONDONE),
    .CHBONDO (user_CHBONDO),
    .PMARXLOCK (user_PMARXLOCK),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRCCCLK (user_RXRCCCLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXERR (user_TXERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
.BREFCLKNIN (user_BREFCLKNIN),
.BREFCLKPIN (user_BREFCLKPIN),
.CHBONDI (user_CHBONDI),
.ENCHANSYNC (user_ENCHANSYNC),
.ENCOMMAALIGN (user_ENCOMMAALIGN),
.ENCPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.PMAINIT (user_PMAINIT),
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGRW (user_PMAREGRW),
.PMARESTROBE (user_PMARESTROBE),
.PMARKLOCKSEL (user_PMARKLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCCLK),
.REFCLK2 (user_REFCCLK2),
.REFCLKBSEL (user_REFCCLKBSEL),
.REFCLKSEL (user_REFCCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXSRCCLK (user_RXUSRCLK),
.RXSRCCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSRCCLK (user_TXUSRCLK),
.TXSRCCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 2;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11’b00101111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11’b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11’b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11’b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b000000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 2;
defparam user_instance_name.CLOCK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLOCK_COR_MAX_LAT = 36;
defparam user_instance_name.CLOCK_COR_MIN_LAT = 28;
defparam user_instance_name.CLOCK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLOCK_COR_SEQ_1_2 = 11'b000000000000;
defparam user_instance_name.CLOCK_COR_SEQ_1_3 = 11'b000000000000;
defparam user_instance_name.CLOCK_COR_SEQ_1_4 = 11'b000000000000;
defparam user_instance_name.CLOCK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLOCK_COR_SEQ_1_USE = "FALSE";
defparam user_instance_name.CLOCK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLOCK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLOCK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b0001111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "25_40";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

Commonly Used Constraints

None
GT10_10GE_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_10GE_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Xilinx protocol 10-gigabit transceiver supports 4 and 8-byte data paths. The letter $n$ represents number of bytes of the data path. Valid values are 4 or 8.

You can also set attributes for the primitives. See the *RocketIO X Transceiver User Guide* for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of $n$. For a description of each of the ports, see the *RocketIO X Transceiver User Guide*. 

---

### GT10_10GE_4

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDONE</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td></td>
</tr>
<tr>
<td>CHBONDI(4:0)</td>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>PMAREQDATAIN(7:0)</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>RXCHARIS(3:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td></td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK2</td>
<td></td>
</tr>
<tr>
<td>REFCLKBSEL</td>
<td>RXDATA(31:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td></td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td></td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXNOTINTABLE(3:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXINCREBT</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXINTDATANWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>RXN</td>
<td>RXRECClk</td>
</tr>
<tr>
<td>RXP</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXRESET</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXKERR(3:0)</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td></td>
</tr>
<tr>
<td>TXBPASS8B10B(3:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARIS(3:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>TXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td></td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATANWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The table includes signals not directly related to the diagram.*

X109441
## GT10_10GE_8

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDONE</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td></td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSIZE(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(7:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td>RXCHARISK(7:0)</td>
</tr>
<tr>
<td>PMAREGROW</td>
<td></td>
</tr>
<tr>
<td>PMARESTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLKCORNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td></td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK2</td>
<td></td>
</tr>
<tr>
<td>REFCLKBSEL</td>
<td>RXDATA(63:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td></td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXDISPERR(7:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td></td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSSCFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXNOTINTABLE(7:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>RXN</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXP</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRUNDISP(7:0)</td>
</tr>
<tr>
<td>RXRESET</td>
<td></td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXERR(7:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARDISPSTRATE(7:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARIDISPSYM(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARISK(7:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXDATA(63:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>TXRUNDISP(7:0)</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT10_10GE_4 VHDL Code

-- Component Attribute specification for GT10_10GE_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_10GE_4
-- synopsys translate_off
generic (
    CHAN_BOND_LIMIT           : integer := 16;
    CHAN_BOND_MODE            : string := "OFF";
    CHAN_BOND_ONE_SHOT        : boolean := FALSE;
    CHAN_BOND_SEQ_1_1         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_2         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_3         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_4         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_1_MASK      : bit_vector := "0000";
    CHAN_BOND_SEQ_2_1         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_2         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_3         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_4         : bit_vector := "00000000000";
    CHAN_BOND_SEQ_2_MASK      : bit_vector := "0000";
    CHAN_BOND_SEQ_2_USE       : boolean := TRUE;
    CHAN_BOND_SEQ_LEN         : integer := 8;
    CHAN_BOND_64B66B_SV       : boolean := FALSE;
    CLK_COR_MAX_LAT           : integer := 36;
    CLK_COR_MIN_LAT           : integer := 28;
    CLK_COR_SEQ_1_1           : bit_vector := "10000011110";
    CLK_COR_SEQ_1_2           : bit_vector := "00000000000";
    CLK_COR_SEQ_1_3           : bit_vector := "00000000000";
    CLK_COR_SEQ_1_4           : bit_vector := "00000000000";
    CLK_COR_SEQ_1_MASK        : bit_vector := "0000";
    CLK_COR_SEQ_2_1           : bit_vector := "00000000000";
    CLK_COR_SEQ_2_2           : bit_vector := "00000000000";
    CLK_COR_SEQ_2_3           : bit_vector := "00000000000";
    CLK_COR_SEQ_2_4           : bit_vector := "00000000000";
    CLK_COR_SEQ_2_MASK        : bit_vector := "0000";
    CLK_COR_SEQ_2_USE         : boolean := TRUE;
    CLK_COR_SEQ_LEN           : integer := 8;
    CLK_CORRECT_USE           : boolean := TRUE;
    PMA_PWR_CNTRL             : bit_vector := "11111111";
    PMA_SPEED                 : string := "6_32";
    RX_BUFFER_USE             : boolean := TRUE;
    RX_LOS_INVALID_INC        : integer := 1;
    RX_LOS_THRESHOLD          : integer := 4;
    RX_LOSS_OF_SYNC_FSM       : boolean := TRUE;
    SH_CNT_MAX                : integer := 64;
    SH_INVALID_CNT_MAX       : integer := 16;
    TX_BUFFER_USE             : boolean := TRUE);
-- synopsys translate_on

port ( 
    BREFCLKNOUT : out std_logic;
    BREFCLKPOUT : out std_logic;
    CHBONDDONE : out std_logic;
    CHBONDO : out std_logic_vector(4 downto 0);
    PMARXLOCK : out std_logic;
    RXBUFSTATUS : out std_logic_vector(1 downto 0);
    RXCHARISCOMMA : out std_logic_vector(3 downto 0);
    RXCHARISK : out std_logic_vector(3 downto 0);
    RXCLKCORCNT : out std_logic_vector(2 downto 0);
    RXCOMMADET : out std_logic;
    RXDATA : out std_logic_vector(31 downto 0);
    RXDISPERR : out std_logic_vector(3 downto 0);
    RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
    RXNOTINTABLE : out std_logic_vector(3 downto 0);
    RXREALIGN : out std_logic;
    RXRECCLK : out std_logic;
    RXRUNDISP : out std_logic_vector(3 downto 0);
    TXBUFERR : out std_logic;
    TXKERR : out std_logic_vector(3 downto 0);
    TXN : out std_logic;
    TXOUTCLK : out std_logic;
    TXP : out std_logic;
    TXRUNDISP : out std_logic_vector(3 downto 0);
    BREFCLKNIN : in std_logic;
    BREFCLKPIN : in std_logic;
    CHBONDI : in std_logic_vector(4 downto 0);
    ENCHANSYNC : in std_logic;
    ENNCOMMAALIGN : in std_logic;
    ENPCOMMAALIGN : in std_logic;
    LOOPBACK : in std_logic_vector(1 downto 0);
    PMAINIT : in std_logic;
    PMAREGADDR : in std_logic_vector(5 downto 0);
    PMAREGDATAIN : in std_logic_vector(7 downto 0);
    PMAREGWR : in std_logic;
    PMAREGSTROBE : in std_logic;
    PMARXLOCKSEL : in std_logic_vector(1 downto 0);
    POWERDOWN : in std_logic;
    REFCLK : in std_logic;
    REFCLK2 : in std_logic;
    REFCLKBSEL : in std_logic;
    REFCLKSEL : in std_logic;
    RXBLOCKSYNC64B66BUSE : in std_logic;
    RXCOMMADETUSE : in std_logic;
    RXDATAWIDTH : in std_logic_vector(1 downto 0);
    RXDEC64B66BUSE : in std_logic;
    RXDEC8B10BUSE : in std_logic;
    RXDESCRAM64B66BUSE : in std_logic;
    RXIGNOREBTF : in std_logic;
    RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
    RXN : in std_logic;
    RXP : in std_logic;
    RXPOLARITY : in std_logic;
    RXRESET : in std_logic;
    RXSLIDE : in std_logic;
    RXUSRCLK : in std_logic;
    RXUSRCLK2 : in std_logic;
)
end component;

-- Component Attribute specification for GT10_10GE_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_10GE_4 should be placed
-- in architecture after the begin keyword

GT10_10GE_4_INSTANCE_NAME : GT10_10GE_4

-- synopsys translate_off

generic map (   CHAN_BOND_LIMIT => integer_value,
                CHAN_BOND_MODE => string_value,
                CHAN_BOND_ONE_SHOT => boolean_value,
                CHAN_BOND_SEQ_1_1 => bit_value,
                CHAN_BOND_SEQ_1_2 => bit_value,
                CHAN_BOND_SEQ_1_3 => bit_value,
                CHAN_BOND_SEQ_1_4 => bit_value,
                CHAN_BOND_SEQ_1_MASK => bit_value,
                CHAN_BOND_SEQ_2_1 => bit_value,
                CHAN_BOND_SEQ_2_2 => bit_value,
                CHAN_BOND_SEQ_2_3 => bit_value,
                CHAN_BOND_SEQ_2_4 => bit_value,
                CHAN_BOND_SEQ_2_MASK => bit_value,
                CHAN_BOND_SEQ_2_USE => boolean_value,
                CHAN_BOND_SEQ_LEN => integer_value,
                CHAN_BOND_64B66B_SV => boolean_value,
                CLK_COR_MAX_LAT => integer_value,
                CLK_COR_MIN_LAT => integer_value,
                CLK_COR_SEQ_1_1 => bit_value,
                CLK_COR_SEQ_1_2 => bit_value,
                CLK_COR_SEQ_1_3 => bit_value,
                CLK_COR_SEQ_1_4 => bit_value,
                CLK_COR_SEQ_1_MASK => bit_value,
                CLK_COR_SEQ_2_1 => bit_value,
                CLK_COR_SEQ_2_2 => bit_value,
                CLK_COR_SEQ_2_3 => bit_value,
                CLK_COR_SEQ_2_4 => bit_value,)
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SH_CNT_MAX => integer_value,
SH_INVALID_CNT_MAX => integer_value,
TX_BUFFER_USE => boolean_value)
-- synopsys translate_on
port map (BREFCLKNOUT => user_BREFCLKNOUT,
BREFCLKPOUT => user_BREFCLKPOUT,
CHBONDDONE => user_CHBONDDONE,
CHBONDO => user_CHBONDO,
PMARXLOCK => user_PMARXLOCK,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCLKCORCTRL => user_RXCLKCORCTRL,
RXCOMMADET => user_RXCOMMADET,
RXDISP => user_RXDATA,
RXERROR => user_RXERROR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLECK => user_RXRECLECK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXOUTCLK => user_TXOUTCLK,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLKIN => user_BREFCLKIN,
BREFCLKPIN => user_BREFCLKPIN,
CHBONDIN => user_CHBONDIN,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPOMCOMMAALIGN => user_ENPOMCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGREGISTRY => user_PMAREGREGISTRY,
PMARXLOCKSEL => user_PMARXLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT10_10GE_8 Code
-- Component Attribute specification for GT10_10GE_8
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_10GE_8
-- synopsys translate_off
generic (
  CHAN_BOND_LIMIT : integer := 16;
  CHAN_BOND_MODE : string := "OFF";
  CHAN_BOND_ONE_SHOT : boolean := FALSE;
  CHAN_BOND_SEQ_1_1 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_1_2 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_1_3 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_1_4 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
  CHAN_BOND_SEQ_2_1 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_2_2 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_2_3 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_2_4 : bit_vector := "0000000000000000";
  CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
  CHAN_BOND_SEQ_2_USE : boolean := TRUE;
  CHAN_BOND_SEQ_LEN : integer := 8;
  CHAN_BOND_64B66B_SV : boolean := FALSE;
  CLK_COR_MAX_LAT : integer := 36;
CLK_COR_MIN_LAT : integer := 28;
CLK_COR_SEQ_1_1 : bit_vector := "10000011110";
CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
CLK_COR_SEQ_1_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
CLK_COR_SEQ_2_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_USE : boolean := TRUE;
CLK_COR_SEQ_DROP : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 8;
CLK_CORRECT_USE : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "11111111";
PMA_SPEED : string := "6_64";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
SH_CNT_MAX : integer := 64;
SH_INVALID_CNT_MAX : integer := 16;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (BREFCLKNOUT : out std_ulogic;
BREFCLKPOUT : out std_ulogic;
CHBONDDONE : out std_ulogic;
CHBONDO : out std_logic_vector(4 downto 0);
PMARXLOCK : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(7 downto 0);
RXCHARISK : out std_logic_vector(7 downto 0);
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXDATA : out std_logic_vector(63 downto 0);
RXDISPERR : out std_logic_vector(7 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(7 downto 0);
RXREALIGN : out std_ulogic;
RXRECECLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(7 downto 0);
TxBFUERR : out std_ulogic;
TXKERR : out std_logic_vector(7 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXF : out std_ulogic;
TXRUNDISP : out std_logic_vector(7 downto 0);
BREFCLKNIN : in std_ulogic;
BREFCLKPIN : in std_ulogic;
CHBONDI : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENFCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAINIT : in std_ulogic;
PMAREGADDR : in std_logic_vector(5 downto 0);
PMAREGDATAIN : in std_logic_vector(7 downto 0);
PMAREGRW : in std_ulogic;
PMAREGSTROBE : in std_ulogic;
PMARXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKBSEL : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXBLOCKSYNC64B66BUSE : in std_ulogic;
RXCOMMADETUSE : in std_ulogic;
RXDATALengthD : in std_logic_vector(1 downto 0);
RXDATAWidth : in std_logic_vector(1 downto 0);
RXENC64B66BUSE : in std_ulogic;
RXENC8B10BUSE : in std_ulogic;
RXGEARBOX64B66BUSE : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(7 downto 0);
TXCHARDISPMODE : in std_logic_vector(7 downto 0);
TXCHARDSPVAL : in std_logic_vector(7 downto 0);
TXCHARISK : in std_logic_vector(7 downto 0);
TXDATA : in std_logic_vector(63 downto 0);
TXDATALengthD : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_ulogic;
TXENC8B10BUSE : in std_ulogic;
TXGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATALengthD : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXSCRAM64B66BUSE : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT10_10GE_8
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_10GE_8 should be placed
-- in architecture after the begin keyword

GT10_10GE_8_INSTANCE_NAME : GT10_10GE_8
-- synopsys translate_off
generic map (  
  CHAN_BOND_LIMIT => integer_value,
  CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2 USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_64B66B_SV => boolean_value,
CLK_COR_MAX_LAT => integer_value,
CLK_COR_MIN_LAT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2 Drop => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INC => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SH_CNT_MAX => integer_value,
SH_INVALID_CNT_MAX => integer_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on

port map (}
BREFCLKOUT => user_BREFCLKOUT,
BREFCLKPOUT => user_BREFCLKPOUT,
CHBOUNDONE => user_CHBOUNDONE,
CHBOND0 => user_CHBOND0,
PMA_RXLOCK => user_PMA_RXLOCK,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXDATA => user_RXDATA,
RXDISPER => user_RXDISPER,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCCLK => user_RXRECCCLK,
Verilog Instantiation Templates

GT10_10GE_4 Verilog Code

GT10_10GE_4 GT10_10GE_4_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .PMAXLOCK (user_PMAXLOCK),
  .RXXBUFSTATUS (user_RXXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECLK (user_RXRECLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFF (user_TXBUFF),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),
  .BREFCLKNIN (user_BREFCLKNIN),
  .BREFCLKPIN (user_BREFCLKPIN),
  .CHBONDI (user_CHBONDI),
  .ENCHANSYNC (user_ENCHANSYNC),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .PMAIN (user_PMAIN),
  .PMAREGADDR (user_PMAREGADDR),
  .PMAREGDATAIN (user_PMAREGDATAIN),
  .PMAREGRW (user_PMAREGRW),
  .PMARESTROBE (user_PMARESTROBE),
  .PMAXLOCKSEL (user_PMAXLOCKSEL),
  .POWERDOWN (userPOWERDOWN),
  .REFCLK (user_REFCLK),
  .REFCLK2 (user_REFCLK2),
  .REFCLKBSEL (user_REFCLKBSEL),
  .REFCLKSEL (user_REFCLKSEL),
  .RXXLOCKSYNC64B66BUSE (user_RXXLOCKSYNC64B66BUSE),
  .RXCOMMADETUSE (user_RXCOMMADETUSE),
  .RXDATAWIDTH (user_RXDATAWIDTH),
  .RXDEC64B66BUSE (user_RXDEC64B66BUSE),
  .RXDEC8B10BUSE (user_RXDEC8B10BUSE),
  .RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
  .RXIGNOREBTF (user_RXIGNOREBTF),
  .RXINTDATAWIDTH (user_RXINTDATAWIDTH),
  .RXN (user_RXN),
  .RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 8;
defparam user_instance_name.CHAN_BOND_64B66B_SV = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b10000011110;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "TRUE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 8;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "6_32";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SH_CNT_MAX = 64;
defparam user_instance_name.SH_INVALID_CNT_MAX = 16;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

GT10_10GE_8 Verilog Code

GT10_10GE_8 GT10_10GE_8_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONO (user_CHBONO),
    .PMARXLOCK (user_PMARXLOCK),
    .RXXBUFSTATUS (user_RXXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECECLK (user_RXRECECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLKNIN (user_BREFCLKNIN),
    .BREFCLKPIN (user_BREFCLKPIN),
    .CHBONDI (user_CHBONDI),
    .ENCHANSYNC (userENCHANSYNC),
    .ENMCOMMAALIGN (userENMCOMMAALIGN),
    .ENPCOMMAALIGN (userENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .PMAINIT (user_PMAINIT),
    .PMAREGADDR (user_PMAREGADDR),
    .PMAREGDATAIN (user_PMAREGDATAIN),
    .PMAREGRW (user_PMAREGRW),
    .PMAREGSTROBE (user_PMAREGSTROBE),
    .PMARXLOCKSEL (user_PMARXLOCKSEL),
    .POWERDOWN (user_POWERDOWN),
    .REFCLK (user_REFCLOCK),
    .REFCLK2 (user_REFCLOCK2),
    .REFCLKBSEL (user_REFCLOCKBSEL),
    .REFCLKSEL (user_REFCLOCKSEL),
    .RXXBLOCKSYNC64B66BUSE (user_RXXBLOCKSYNC64B66BUSE),
    .RXCOMMADETUSE (user_RXCOMMADETUSE),
    .RXDATAWIDTH (user_RXDATAWIDTH),
    .RXDEC64B66BUSE (user_RXDEC64B66BUSE),
    .RXDEC8B10BUSE (user_RXDEC8B10BUSE),
    .RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
    .RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXX (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISP (user_TXCHARDISP),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_1_USE = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "TRUE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 8;
defparam user_instance_name.CHAN_BOND_64B66B_SV = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b10000011110;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "TRUE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 8;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "6_64";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.SH_CNT_MAX = 64;
defparam user_instance_name.SH_INVALID_CNT_MAX = 16;
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
GT10_10GFC_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Xilinx protocol 10-gigabit transceiver supports 4 and 8-byte data paths. The letter $n$ represents number of bytes of the data path. Valid values are 4 or 8.

You can also set attributes for the primitives. See the RocketIO X Transceiver User Guide for a description of these attributes and their default attribute values.

For a description of the input and output ports for all values of $n$, see the RocketIO X Transceiver User Guide.
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT10_10GFC_4 VHDL Code

-- Component Attribute specification for GT10_10GFC_4
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT10_10GFC_4
  -- synopsys translate_off
gENERIC (  
    CHAN_BOND_LIMIT : integer := 16;
    CHAN_BOND_MODE  : string := "OFF";
    CHAN_BOND_ONE_SHOT  : boolean := FALSE;
    CHAN_BOND_SEQ_1_1  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_2  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_3  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_4  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_1  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_2  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_3  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_4  : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_USE : boolean := TRUE;
    CHAN_BOND_SEQ_LEN : integer := 8;
    CHAN_BOND_64B66B_SV : boolean := FALSE;
    CLK_COR_MAX_LAT : integer := 36;
    CLK_COR_MIN_LAT : integer := 28;
    CLK_COR_SEQ_1_1  : bit_vector := "1000000000000000";
    CLK_COR_SEQ_1_2  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_1_3  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_1_4  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_1_MASK : bit_vector := "0000";
    CLK_COR_SEQ_2_1  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_2  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_3  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_4  : bit_vector := "0000000000000000";
    CLK_COR_SEQ_2_MASK : bit_vector := "0000";
    CLK_COR_SEQ_2_USE : boolean := TRUE;
    CLK_COR_SEQ_DROP : boolean := FALSE;
    CLK_COR_SEQ_LEN : integer := 8;
    CLK_CORRECT_USE : boolean := TRUE;
    PMA_PWR_CNTRL : bit_vector := "11111111";
    PMA_SPEED : string := "0_32";
    RX_BUFFER_USE : boolean := TRUE;
    RX_LOS_INVALID_INCR : integer := 1;
    RX_LOS_THRESHOLD : integer := 4;
    RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
    SH_CNT_MAX : integer := 64;
    SH_INVALID_CNT_MAX : integer := 16;
    TX_BUFFER_USE : boolean := TRUE);
-- synopsys translate_on

port ( 
    BREFCLKNOUT               : out std_logic;
    BREFCLKPOUT               : out std_logic;
    CHBONDDONE                : out std_logic;
    CHBONDID                   : out std_logic_vector(4 downto 0);
    PMARXLOCK                 : out std_logic;
    RXBUFFSTATUS               : out std_logic_vector(1 downto 0);
    RXCHARISCOMMA              : out std_logic_vector(3 downto 0);
    RXCHARISK                  : out std_logic_vector(3 downto 0);
    RXCLKCORCNT                : out std_logic_vector(2 downto 0);
    RXCOMMADET                  : out std_logic;
    RXDATA                     : out std_logic_vector(31 downto 0);
    RXDISPERR                  : out std_logic_vector(3 downto 0);
    RXLOSSOFSYNC               : out std_logic_vector(1 downto 0);
    RXNOTINTABLE               : out std_logic_vector(3 downto 0);
    RXREALIGN                  : out std_logic;
    RXRECLK                    : out std_logic;
    RXRUNDISP                  : out std_logic_vector(3 downto 0);
    TXBUFDERR                   : out std_logic;
    TXKERR                     : out std_logic_vector(3 downto 0);
    TXN                        : out std_logic;
    TXOUTCLK                   : out std_logic;
    TXP                        : out std_logic;
    TXRUNDISP                  : out std_logic_vector(3 downto 0);
    BREFCLKNIN                 : in std_logic;
    BREFCLKPIN                 : in std_logic;
    CHBONDI                    : in std_logic_vector(4 downto 0);
    ENCHANSYNC                  : in std_logic;
    ENNCOMMAALIGN               : in std_logic;
    ENPCOMMAALIGN               : in std_logic;
    LOOPBACK                   : in std_logic_vector(1 downto 0);
    PMAINIT                    : in std_logic;
    PMAREGADDR                 : in std_logic_vector(5 downto 0);
    PMAREGDATAIN               : in std_logic_vector(7 downto 0);
    PMAREGRW                   : in std_logic;
    PMAREGSTROBE               : in std_logic;
    PMARXLOCKSEL               : in std_logic_vector(1 downto 0);
    POWERDOWN                  : in std_logic;
    REFCLK                     : in std_logic;
    REFCLK2                    : in std_logic;
    REFCLKBSSEL                : in std_logic;
    REFCLKSEL                  : in std_logic;
    RXBLOCKSYNC64B66BUSE        : in std_logic;
    RXCOMMADETUSE              : in std_logic;
    RXDATAWIDTH                 : in std_logic_vector(1 downto 0);
    RXDECC64B66BUSE            : in std_logic;
    RXDECC8B10BUSE              : in std_logic;
    RXDESCRAM64B66BUSE          : in std_logic;
    RXIGNOREBF                  : in std_logic;
    RXINTDATAWIDTH              : in std_logic_vector(1 downto 0);
    RXN                        : in std_logic;
    RXP                        : in std_logic;
    RXPOLARITY                 : in std_logic;
    RXRESET                    : in std_logic;
    RXSLIDE                    : in std_logic;
    RXUSRCLK                   : in std_logic;
    RXUSRCLK2                  : in std_logic;
);
end component;

-- Component Attribute specification for GT10_10GFC_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_10GFC_4 should be placed
-- in architecture after the begin keyword

GT10_10GFC_4_INSTANCE_NAME : GT10_10GFC_4
-- synopsys translate_off
generic map (    CHAN_BOND_LIMIT => integer_value,
                 CHAN_BOND_MODE => string_value,
                 CHAN_BOND_ONE_SHOT => boolean_value,
                 CHAN_BOND_SEQ_1_1 => bit_value,
                 CHAN_BOND_SEQ_1_2 => bit_value,
                 CHAN_BOND_SEQ_1_3 => bit_value,
                 CHAN_BOND_SEQ_1_4 => bit_value,
                 CHAN_BOND_SEQ_1_MASK => bit_value,
                 CHAN_BOND_SEQ_2_1 => bit_value,
                 CHAN_BOND_SEQ_2_2 => bit_value,
                 CHAN_BOND_SEQ_2_3 => bit_value,
                 CHAN_BOND_SEQ_2_4 => bit_value,
                 CHAN_BOND_SEQ_2_MASK => bit_value,
                 CHAN_BOND_SEQ_2_USE => boolean_value,
                 CHAN_BOND_SEQ_LEN => integer_value,
                 CHAN_BOND_64B66B_SV => boolean_value,
                 CLK_COR_MAX_LAT => integer_value,
                 CLK_COR_MIN_LAT => integer_value,
                 CLK_COR_SEQ_1_1 => bit_value,
                 CLK_COR_SEQ_1_2 => bit_value,
                 CLK_COR_SEQ_1_3 => bit_value,
                 CLK_COR_SEQ_1_4 => bit_value,
                 CLK_COR_SEQ_1_MASK => bit_value,
                 CLK_COR_SEQ_2_1 => bit_value,
                 CLK_COR_SEQ_2_2 => bit_value,
                 CLK_COR_SEQ_2_3 => bit_value,
                 CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SH_CNT_MAX => integer_value,
SH_INVALID_CNT_MAX => integer_value,
TX_BUFFER_USE => boolean_value)

-- synopsys translate_on
port map ( 
  BREFCLKNOUT => user_BREFCLKNOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDDONE => user_CHBONDDONE,
  CHBONO => user_CHBONO,
  PMARXLOCK => user_PMARXLOCK,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCLKCORCMA => user_RXCLKCORCMA,
  RXCOMMADET => user_RXCOMMADET,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECLK => user_RXRECLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFERR => user_TXBUFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXOUTCLK => user_TXOUTCLK,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLKNIN => user_BREFCLKNIN,
  BREFCLKPIN => user_BREFCLKPIN,
  CHBOND => user_CHBOND,
  ENCHANSYNC => user_ENCHANSYNC,
  ENMCOMMAALIGN => user_ENMCOMMAALIGN,
  ENPCOMMAALIGN => user_ENPCOMMAALIGN,
  LOOPBACK => user_LOOPBACK,
  PMAINI => user_PMAINI,
  PMAREGADDR => user_PMAREGADDR,
  PMAREGDATAIN => user_PMAREGDATAIN,
  PMAREGRW => user_PMAREGRW,
  PMAREGSTROBE => user_PMAREGSTROBE,
  PMARXLOCKSEL => user_PMARXLOCKSEL,
  POWERDOWN => user_POWERDOWN,
  REFKLC => user_REFKLC,
  REFKL2 => user_REFKL2,
  REFKLBSEL => user_REFKLBSEL,
  REFKLSSEL => user_REFKLSSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT10_10GFC_8 VHDL Code

-- Component Attribute specification for GT10_10GE_8
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_10GE_8
  -- synopsys translate_off
generic (  
    CHAN_BOND_LIMIT : integer := 16;
    CHAN_BOND_MODE : string := "OFF";
    CHAN_BOND_ONE_SHOT : boolean := FALSE;
    CHAN_BOND_SEQ_1_1 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_2 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_3 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_4 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_1 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_2 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_3 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_4 : bit_vector := "0000000000000000";
    CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
    CHAN_BOND_SEQ_2_USE : boolean := TRUE;
    CHAN_BOND_SEQ_LEN : integer := 8;
    CHAN_BOND_64B66B_SV : boolean := FALSE;
    CLK_COR_MAX_LAT : integer := 36;
  );
CLX_COR_MIN_LAT: integer := 28;
CLX_COR_SEQ_1_1: bit_vector := "10000011110";
CLX_COR_SEQ_1_2: bit_vector := "00000000000";
CLX_COR_SEQ_1_3: bit_vector := "00000000000";
CLX_COR_SEQ_1_4: bit_vector := "00000000000";
CLX_COR_SEQ_1_MASK: bit_vector := "0000";
CLX_COR_SEQ_2_1: bit_vector := "00000000000";
CLX_COR_SEQ_2_2: bit_vector := "00000000000";
CLX_COR_SEQ_2_3: bit_vector := "00000000000";
CLX_COR_SEQ_2_4: bit_vector := "00000000000";
CLX_COR_SEQ_2_MASK: bit_vector := "0000";
CLX_COR_SEQ_2_USE: boolean := TRUE;
CLX_COR_SEQ_DROP: boolean := FALSE;
CLX_COR_SEQ_LEN: integer := 8;
CLX_CORRECT_USE: boolean := TRUE;
PMA_PWR_CNTRL: bit_vector := "11111111";
PMA_SPEED: string := "6_64";
RX_BUFFER_USE: boolean := TRUE;
RX_LOS_INVALID_INCR: integer := 1;
RX_LOS_THRESHOLD: integer := 4;
RX_LOSS_OF_SYNC_FSM: boolean := TRUE;
SH_CNT_MAX: integer := 64;
SH_INVALID_CNT_MAX: integer := 16;
TX_BUFFER_USE: boolean := TRUE);

-- synopsys translate_on
port (BREFCLKNOUT: out std_ulogic;
BREFCLKPOUT: out std_ulogic;
CHBONDDONE: out std_ulogic;
CHBONDO: out std_logic_vector(4 downto 0);
PMA_RXLOCK: out std_ulogic;
RXBFSTATUS: out std_logic_vector(1 downto 0);
RXCHARISCOMMA: out std_logic_vector(7 downto 0);
RXCLKCORCNT: out std_logic_vector(2 downto 0);
RXCOMMADET: out std_ulogic;
RXDATA: out std_logic_vector(63 downto 0);
RXDISPERR: out std_logic_vector(7 downto 0);
RXLOSSOFSYNC: out std_logic_vector(1 downto 0);
RXNOTINTABLE: out std_logic_vector(7 downto 0);
RXREALIGN: out std_ulogic;
RXRECLK: out std_ulogic;
RXRUNDISP: out std_logic_vector(7 downto 0);
TXBUFFERR: out std_ulogic;
TXKERR: out std_logic_vector(7 downto 0);
TXN: out std_ulogic;
TXOUTCLK: out std_ulogic;
TXF: out std_ulogic;
TXRUNDISP: out std_logic_vector(7 downto 0);
BREFCLKIN: in std_ulogic;
BREFCLKPIN: in std_ulogic;
CHBONDI: in std_logic_vector(4 downto 0);
ENCHANSYNC: in std_ulogic;
ENKCOMMAALIGN: in std_ulogic;
ENPCOMMAALIGN: in std_ulogic;
LOOPBACK: in std_logic_vector(1 downto 0);
PMAINIT: in std_ulogic;
end component;

-- Component Attribute specification for GT10_10GE_8
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_10GE_8 should be placed
-- in architecture after the begin keyword

GT10_10GE_8_INSTANCE_NAME : GT10_10GE_8
-- synopsys translate off
generic map
    (CHAN_BOND_LIMIT => integer_value,
     CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CHAN_BOND_64B66B_SV => boolean_value,
CLK_COR_MAX_LAT => integer_value,
CLK_COR_MIN_LAT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_DROP => boolean_value,
CLK_COR_SEQ_LEN => integer_value,
CLK_CORRECT_USE => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
SH_CNT_MAX => integer_value,
SH_INVALID_CNT_MAX => integer_value,
TX_BUFFER_USE => boolean_value

-- synopsys translate_on
port map (}
BREFCLKNOUT => user_BREFCLKNOUT,
BREFCLKPOUT => user_BREFCLKPOUT,
CHBONDDONE => user_CHBONDDONE,
CHBONDO => user_CHBONDO,
PMARXLOCK => user_PMARXLOCK,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCCLK => user_RXRECCCLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXOUTCLK => user_TXOUTCLK,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLKNIN => user_BREFCLKNIN,
BREFCLKPIN => user_BREFCLKPIN,
CHBONDI => user_CHBONDI,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMARXLOCKSEL => user_PMARXLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);
Commonly Used Constraints

None
GT10_OC48_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_OC48_n</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitivo*</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Xilinx protocol 10-gigabit transceiver supports 1, 2 and 4-byte data paths. The letter n represents number of bytes of the data path. Valid values are 1, 2, or 4.

You can also set attributes for the primitives. See the RocketIO X Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of n. For a description of each of the ports, see the RocketIO X Transceiver User Guide.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDONE</td>
</tr>
<tr>
<td>BREFCLKIN</td>
<td></td>
</tr>
<tr>
<td>CHBONDONE(4:0)</td>
<td>CHBONDONE(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSIZE(1:0)</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td></td>
</tr>
<tr>
<td>PMAINIT</td>
<td></td>
</tr>
<tr>
<td>PMAINIT</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(0:0)</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td></td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td>RXCHARISCOMMA(0:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td></td>
</tr>
<tr>
<td>PMAREGROW</td>
<td>RXCHARISCOMMA(0:0)</td>
</tr>
<tr>
<td>PMAREGROW</td>
<td></td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td>RXCHARISCOMMA(0:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td></td>
</tr>
<tr>
<td>POWERDOW</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>POWERDOW</td>
<td></td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK</td>
<td></td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXDATA(7:0)</td>
</tr>
<tr>
<td>REFCLK2</td>
<td></td>
</tr>
<tr>
<td>REFCLKSELE</td>
<td>RXDISERR(0:0)</td>
</tr>
<tr>
<td>REFCLKSELE</td>
<td></td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXDISERR(0:0)</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>RXDISERR(0:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td></td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>RXLOSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXNOTINTABLE(0:0)</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td></td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RXRUNDISP(0:0)</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>RXIP</td>
<td>RXRUNDISP(0:0)</td>
</tr>
<tr>
<td>RXIP</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRUNDISP(0:0)</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>RXRESET</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXRESET</td>
<td></td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXKERR(0:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(0:0)</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td></td>
</tr>
<tr>
<td>TXBYPASS8B10B(0:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXBYPASS8B10B(0:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHAR0DISPMODE(0:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHAR0DISPMODE(0:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARISK(0:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXCHARISK(0:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(7:3)</td>
<td>TXREALIGN</td>
</tr>
<tr>
<td>TXDATA(7:3)</td>
<td></td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td></td>
</tr>
<tr>
<td>TXXGEARBOX64B66BUSE</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXXGEARBOX64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
<tr>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLKIN</td>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANDSYNC</td>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>PMARKLOCK(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK2</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCCLKBSEL</td>
<td>RXDATA(15:0)</td>
</tr>
<tr>
<td>REFCCLKSEL</td>
<td>RXDATA(15:0)</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXDISPER(1:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>RXDISPER(1:0)</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXNOTINTABLE(1:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td>RXNOTINTABLE(1:0)</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXN</td>
<td>RXRECECLK</td>
</tr>
<tr>
<td>RXP</td>
<td>RXRECECLK</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRUNDISP(1:0)</td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRUNDISP(1:0)</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXERRR(1:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(1:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPV(1:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXGEBROBOX64B66BUSE</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>GT10_OC48_4</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDONE</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td></td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
<td>CHBONDQ(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>PMAIN</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td></td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>RXCHARISK(3:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td></td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK2</td>
<td></td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXDATA(31:0)</td>
</tr>
<tr>
<td>REFLKSEL</td>
<td></td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RDXDISPERR(3:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td></td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDEC68B10BUSE</td>
<td>RXNOTINTABLE(3:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXINTDATANWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>RXN</td>
<td>RXRECCCLK</td>
</tr>
<tr>
<td>RXP</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXRESET</td>
<td></td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXKERR(3:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARDISPMODE(3:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXCHARDISPVAL(3:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARISK(3:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
<td>TNP</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXENC68B10BUSE</td>
<td>TXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXGEARBOX64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATANWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT10_OC48_1 VHDL Code

-- Component Attribute specification for GT10_OC48_1
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_OC48_1
-- synopsys translate_off
generic(
    ALIGN_COMMA_WORD          : integer := 1;
    COMMA_10B_MASK            : bit_vector := "0011111111";
    DEC_MCOMMA_DETECT         : boolean := TRUE;
    DEC_PCOMMA_DETECT         : boolean := TRUE;
    MCOMMA_10B_VALUE          : bit_vector := "0010101010";
    MCOMMA_DETECT             : boolean := TRUE;
    PCOMMA_10B_VALUE          : bit_vector := "0010101010";
    PCOMMA_DETECT             : boolean := TRUE;
    PMA_PWR_CNTRL             : bit_vector := "11111111";
    PMA_SPEED                 : string := "31_8";
    RX_BUFFER_USE             : boolean := TRUE;
    RX_LOS_INVALID_INCR       : integer := 1;
    RX_LOS_THRESHOLD          : integer := 4;
    RX_LOSS_OF_SYNC_FSM       : boolean := TRUE;
    TX_BUFFER_USE             : boolean := TRUE);
-- synopsys translate_on
port(
    BREFCLKNOUT               : out std_ulogic;
    BREFCLKPOUT               : out std_ulogic;
    CHBONDDONE                : out std_logic;
    CHBONDO                   : out std_logic_vector(4 downto 0);
    PMARXLOCK                 : out std_ulogic;
    RXBUFFSTATUS              : out std_logic_vector(1 downto 0);
    RXCHARISCOMMA             : out std_logic_vector(0 downto 0);
    RXCHARISK                 : out std_logic_vector(0 downto 0);
    RXCLKCORCNT               : out std_logic_vector(2 downto 0);
    RXCOMMADET                : out std_logic;
    RXDATA                    : out std_logic_vector(7 downto 0);
    RXDISPERR                 : out std_logic_vector(0 downto 0);
    RXLOSSOFSYNC              : out std_logic_vector(1 downto 0);
    RXNOTINTABLE              : out std_logic_vector(0 downto 0);
    RXREALIGN                 : out std_ulogic;
    RXRECLK                   : out std_logic;
    RXRUNDISP                 : out std_logic_vector(0 downto 0);
    TXBUFFERR                 : out std_logic;
    TXKERR                    : out std_logic_vector(0 downto 0);
    TXN                       : out std_logic;
    TXOUTCLK                  : out std_logic;
    TXP                       : out std_logic;
    TXRUNDISP                 : out std_logic_vector(0 downto 0);
    BREFCLKNIN                : in std_logic;
end component;

-- Component Attribute specification for GT10_OC48_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_OC48_1 should be placed
-- in architecture after the begin keyword

GT10_OC48_1_INSTANCE_NAME : GT10_OC48_1
-- synopsys translate_off
generic map (
  ALIGN_COMMA_WORD          => integer_value,
  COMMA_10B_MASK            => bit_value,
  DEC_MCOMMA_DETECT         => boolean_value,
  DEC_PCOMMA_DETECT         => boolean_value,
  MCOMMA_10B_VALUE          => bit_value,
  MCOMMA_DETECT             => boolean_value,
  PCOMMA_10B_VALUE          => bit_value,
  PCOMMA_DETECT             => boolean_value,
  PMA_PWR_CNTRL             => bit_value,
  PMA_SPEED                 => string_value,
  RX_BUFFER_USE             => boolean_value,
  RX_LOS_INVALID_INCR       => integer_value,
  RX_LOS_THRESHOLD          => integer_value,
  RX_LOSS_OF_SYNC_FSM       => boolean_value,
  TX_BUFFER_USE             => boolean_value)
-- synopsys translate_on
port map (
  BREFCLKNOUT               => user_BREFCLKNOUT,
  BREFCLKPOUT               => user_BREFCLKPOUT,
  CHBONDDONE                => user_CHBONDDONE,
  CHBONDO                   => user_CHBONDO,
  PMARXLOCK                 => user_PMARXLOCK,
  RXBUFSTATUS               => user_RXBUFSTATUS,
  RXCHARISCOMMA             => user_RXCHARISCOMMA,
  RXCHARISK                 => user_RXCHARISK,
  RXCLKCORCNT               => user_RXCLKCORCNT,
  RXCOMMADET                => user_RXCOMMADET,
  RXDATA                    => user_RXDATA,
  RXDISPERR                 => user_RXDISPERR,
  RXLOSSOFSYNC              => user_RXLOSSOFSYNC,
  RXNOTINTABLE              => user_RXNOTINTABLE,
  RXREALIGN                 => user_RXREALIGN,
  RXRECLECLK                => user_RXRECLECLK,
  RXRUNDISP                 => user_RXRUNDISP,
  TXBUFFERR                 => user_TXBUFFERR,
  TXKERR                    => user_TXKERR,
  TXN                       => user_TXN,
  TXOUTCLK                  => user_TXOUTCLK,
  TXP                       => user_TXP,
  TXRUNDISP                 => user_TXRUNDISP,
  BREFCLKNIN                => user_BREFCLKNIN,
  BREFCLKPIN                => user_BREFCLKPIN,
  CHBONDI                   => user_CHBONDI,
  ENCHANSYNC                => user_ENCHANSYNC,
  ENMCOMMAALIGN             => user_ENMCOMMAALIGN,
  ENPCOMMAALIGN             => user_ENPCOMMAALIGN,
  LOOPBACK                  => user_LOOPBACK,
  PMAINIT                   => user_PMAINIT,
  PMAREGADDR                => user_PMAREGADDR,
  PMAREGDATAIN              => user_PMAREGDATAIN,
  PMAREGRW                  => user_PMAREGRW,
  PMAREGSTROBE              => user_PMAREGSTROBE,
PMARXLOCKSEL  => user_PMARXLOCKSEL,
POWERDOWN     => user_POWERDOWN,
REFCLK        => user_REFCLK,
REFCLK2       => user_REFCLK2,
REFCLKBSEL    => user_REFCLKBSEL,
REFCLKSEL     => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH   => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF   => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN           => user_RXN,
RXP           => user_RXP,
RXPOLARITY    => user_RXPOLARITY,
RXRESET       => user_RXRESET,
RXSLIDE       => user_RXSLIDE,
RXUSRCLK      => user_RXUSRCLK,
RXUSRCLK2     => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK     => user_TXCHARISK,
TXDATA        => user_TXDATA,
TXDATAWIDTH   => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT     => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY    => user_TXPOLARITY,
TXRESET       => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK      => user_TXUSRCLK,
TXUSRCLK2     => user_TXUSRCLK2);

GT10_OC48_2 VHDL Code
-- Component Attribute specification for GT10_OC48_2
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_OC48_2
-- synopsys translate_off
generic (ALIGN_COMMA_WORD : integer := 1;
            COMMA_10B_MASK : bit_vector := "0011111111";
            DEC_MCOMMA_DETECT : boolean := TRUE;
            DEC_PCOMMA_DETECT : boolean := TRUE;
            MCOMMA_10B_VALUE : bit_vector := "0010101010";
            MCOMMA_DETECT : boolean := TRUE;
            PCOMMA_10B_VALUE : bit_vector := "0010101010";
            PCOMMA_DETECT : boolean := TRUE;
            PMA_PWR_CNTRL : bit_vector := "11111111";
            PMA_SPEED : string := "31_16";
            RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE;

-- synopsys translate_on

port (  
BREFCLKNOUT : out std_ulogic;
BREFCLKPOUT : out std_ulogic;
CHBONDDONE : out std_ulogic;
CHBOND0 : out std_logic_vector(4 downto 0);
PMARXLOCK : out std_ulogic;
RXBUFSTATUS : out std_logic_vector(1 downto 0);
RXCHARISCOMMA : out std_logic_vector(1 downto 0);
RXCHARISK : out std_logic_vector(1 downto 0);
RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXDATA : out std_logic_vector(15 downto 0);
RXDISPERR : out std_logic_vector(1 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(1 downto 0);
RXREALIGN : out std_ulogic;
RXRECLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(1 downto 0);
TXBUFERR : out std_logic;
TXKERR : out std_logic_vector(1 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(1 downto 0);
BREFCLKNIN : in std_ulogic;
BREFCLKPIN : in std_ulogic;
CHBONDI : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAINIT : in std_ulogic;
PMAREGADDR : in std_logic_vector(5 downto 0);
PMAREGDATAIN : in std_logic_vector(7 downto 0);
PMAREGRW : in std_ulogic;
PMAREGSTROBE : in std_ulogic;
PMARXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKBSEL : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXBLOCKSYNC64B66BUSE : in std_ulogic;
RXCOMMADETUSE : in std_ulogic;
RXDATAWIDTH : in std_logic_vector(1 downto 0);
RXDEC64B66BUSE : in std_ulogic;
RXDEC8B10BUSE : in std_ulogic;
RXDESCRAM64B66BUSE : in std_ulogic;
RXIGNOREBTF : in std_ulogic;
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_logic;
RXRESET : in std_logic;
RXSLIDE : in std_logic;
RXUSRCLK : in std_logic;
RXUSRCLK2 : in std_logic;
TXBYPASS8B10B : in std_logic_vector(1 downto 0);
TXCHARDISP_MODE : in std_logic_vector(1 downto 0);
TXCHARDISPVAL : in std_logic_vector(1 downto 0);
TXCHARISK : in std_logic_vector(1 downto 0);
TXDATA : in std_logic_vector(15 downto 0);
TXDATA_WIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_logic;
TXENC8B10BUSE : in std_logic;
TXGEARBOX64B66BUSE : in std_logic;
TXINHIBIT : in std_logic;
TXINTDATA_WIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_logic;
TXRESET : in std_logic;
TXSCRAM64B66BUSE : in std_logic;
TXUSRCLK : in std_logic;
TXUSRCLK2 : in std_logic);

end component;

-- Component Attribute specification for GT10_OC48_2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_OC48_2 should be placed
-- in architecture after the begin keyword

GT10_OC48_2_INSTANCE_NAME : GT10_OC48_2
-- synopsys translate_off
generic map (
  ALIGN_COMMA_WORD => integer_value,
  COMMA_10B_MASK => bit_value,
  DEC_MCOMMA_DETECT => boolean_value,
  DEC_PCOMMA_DETECT => boolean_value,
  MCOMMA_10B_VALUE => bit_value,
  MCOMMA_DETECT => boolean_value,
  PCOMMA_10B_VALUE => bit_value,
  PCOMMA_DETECT => boolean_value,
  PMA_PWR_CNTRL => bit_value,
  PMA_SPEED => string_value,
  RX_BUFFER_USE => boolean_value,
  RX_LOS_INVALID_INCR => integer_value,
  RX_LOS_THRESHOLD => integer_value,
  RX_LOSS_OF_SYNC_FSM => boolean_value,
  TX_BUFFER_USE => boolean_value)
-- synopsys translate_on
port map (
  BREFCLKNOUT => user_BREFCLKNOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDDONE => user_CHBONDDONE,
  CHBONO => user_CHBONO,
PMARXLOCK => user_PMARXLOCK,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXERR => user_TXERR,
TXN => user_TXN,
TXOUTCLK => user_TXOUTCLK,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLKNIN => user_BREFCLKNIN,
BREFCLKPIN => user_BREFCLKPIN,
CHBONDI => user_CHBONDI,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMARXLOCKSEL => user_PMARXLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSSEL => user_REFCLKSSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
GT10_OC48_4 VHDL Code

-- Component Attribute specification for GT10_OC48_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_OC48_4
  generic (ALIGN_COMMA_WORD: integer := 1;
            COMMA_10B_MASK: bit_vector := "0011111111";
            DEC_MCOMMA_DETECT: boolean := TRUE;
            DEC_PCOMMA_DETECT: boolean := TRUE;
            MCOMMA_10B_VALUE: bit_vector := "0010101010";
            MCOMMA_DETECT: boolean := TRUE;
            PCOMMA_10B_VALUE: bit_vector := "0010101010";
            PCOMMA_DETECT: boolean := TRUE;
            PMA_PWR_CNTRL: bit_vector := "11111111";
            PMA_SPEED: string := "31_32";
            RX_BUFFER_USE: boolean := TRUE;
            RX_LOS_INVALID_INCR: integer := 1;
            RX_LOS_THRESHOLD: integer := 4;
            RX_LOSS_OF_SYNC_FSM: boolean := TRUE;
            TX_BUFFER_USE: boolean := TRUE)

port (BREFCLKNOUT: out std_ulogic;
      BREFCLKPOUT: out std_ulogic;
      CHBONDDONE: out std_logic;
      CHBONO: out std_logic;
      PMARXLOCK: out std_ulogic;
      RXBUFSTATUS: out std_logic_vector(1 downto 0);
      RXCHARISCOMMA: out std_logic_vector(3 downto 0);
      RXCHARISK: out std_logic_vector(3 downto 0);
      RXCLKCORCNT: out std_logic_vector(2 downto 0);
      RXCOMMADET: out std_logic;
      RXDATA: out std_logic_vector(31 downto 0);
      RXDISPERR: out std_logic_vector(3 downto 0);
      RXLOSSOFSYNC: out std_logic_vector(1 downto 0);
      RXNOTINTABLE: out std_logic_vector(3 downto 0);
      RXREALIGN: out std_logic;
      RXRECCCLK: out std_logic;
      RXRUNDISP: out std_logic_vector(3 downto 0);
      TXBVFERR: out std_logic;
      TXKERR: out std_logic_vector(3 downto 0);
      TXN: out std_logic;
GT10_OC48_n

end component;

-- Component Attribute specification for GT10_OC48_4
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter constraints here

-- Component Instantiation for GT10_OC48_4 should be placed
-- in architecture after the begin keyword

GT10_OC48_4_INSTANCE_NAME : GT10_OC48_4
-- synopsys translate_off
generic map (
    ALIGN_COMMA_WORD          => integer_value,
    COMMA_10B_MASK            => bit_value,
    DEC_MCOMMA_DETECT         => boolean_value,
    DEC_PCOMMA_DETECT         => boolean_value,
    MCOMMA_10B_VALUE          => bit_value,
    MCOMMA_DETECT             => boolean_value,
    PCOMMA_10B_VALUE          => bit_value,
    PCOMMA_DETECT             => boolean_value,
    PMA_PWR_CNTRL             => bit_value,
    PMA_SPEED                 => string_value,
    RX_BUFFER_USE             => boolean_value,
    RX_LOS_INVALID_INCR       => integer_value,
    RX_LOS_THRESHOLD          => integer_value,
    RX_LOSS_OF_SYNC_FSM       => boolean_value,
    TX_BUFFER_USE             => boolean_value)
-- synopsys translate_on

port map (
    BREFCLKNOUT               => user_BREFCLKNOUT,
    BREFCLKPOUT               => user_BREFCLKPOUT,
    CHBONDDONE                => user_CHBONDDONE,
    CHBONDO                   => user_CHBONDO,
    PMARXLOCK                 => user_PMARXLOCK,
    RXBUFSTATUS               => user_RXBUFSTATUS,
    RXCHARISCOMMA             => user_RXCHARISCOMMA,
    RXCHARISK                 => user_RXCHARISK,
    RXCLKCORCNT               => user_RXCLKCORCNT,
    RXCOMMADET                => user_RXCOMMADET,
    RXDATA                    => user_RXDATA,
    RXDISPERR                 => user_RXDISPERR,
    RXLOSSOFSYNC              => user_RXLOSSOFSYNC,
    RXNOTINTABLE              => user_RXNOTINTABLE,
    RXREALIGN                 => user_RXREALIGN,
    RXRECCCLK                 => user_RXRECCCLK,
    RXRUNDISP                 => user_RXRUNDISP,
    TXBUFERR                  => user_TXBUFERR,
    TXKERR                    => user_TXKERR,
    TXN                       => user_TXN,
    TXOUTCLK                  => user_TXOUTCLK,
    TXP                       => user_TXP,
    TXRUNDISP                 => user_TXRUNDISP,
    BREFCLKNIN                => user_BREFCLKNIN,
    BREFCLKPIN                => user_BREFCLKPIN,
    CHBONDI                   => user_CHBONDI,
    ENCHANSYNC                => user_ENCHANSYNC,
    ENMCOMMAALIGN             => user_ENMCOMMAALIGN,
    ENPCOMMAALIGN             => user_ENPCOMMAALIGN,
    LOOPBACK                  => user_LOOPBACK,
    PMAINIT                   => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMARXLOCKSEL => user_PMARXLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPPVAL => user_TXCHARDISPPVAL,
TXCHARISK => user_TXCHARISK,
TDATA => user_TDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

Verilog Instantiation Templates

GT10_OC48_1 Verilog Code

GT10_OC48_1 GT10_OC48_1_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONDO (user_CHBONDO),
  .PMARXLOCK (user_PMARXLOCK),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
.RXDATA (user_RXDATA),
.RXDISPERR (user_RXDISPERR),
.RXLOSSOFSYNC (user_RXLOSSOFSYNC),
.RXNOTINTABLE (user_RXNOTINTABLE),
.RXREALIGN (user_RXREALIGN),
.RXRECCLOCK (user_RXRECCLOCK),
.RXRUNDISP (user_RXRUNDISP),
.TXBUFERR (user_TXBUFERR),
.TXKERR (user_TXKERR),
.TXN (user_TXN),
.TXOUTCLK (user_TXOUTCLK),
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLKVIN (user_BREFCLKVIN),
.BREFCLKPIN (user_BREFCLKPIN),
.CHBONDI (user_CHBONDI),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.PMAINIT (user_PMAINIT),
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGRW (user_PMAREGRW),
.PMAREGSTROBE (user_PMAREGSTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAn (user_RXINTDATAn),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCCLK (user_RXUSRCCLK),
.RXUSRCCLK2 (user_RXUSRCCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARISMP (user_TXCHARISMP),
.TXCHARISMP (user_TXCHARISMP),
.TXDATA (user_TXDATA),
.TXDATAn (user_TXDATAn),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAn (user_TXINTDATAn),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.COMMA_10B_MASK = 10'b0011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "31_8";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

GT10_OC48_2 Verilog Code

GT10_OC48_2 GT10_OC48_2_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .PMARKLOCK (user_PMARKLOCK),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),

    .BREFCLKNIN (user_BREFCLKNIN),
    .BREFCLKPIN (user_BREFCLKPIN),
    .CHBONDI (user_CHBONDI),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .PMAINIT (user_PMAINIT),
    .PMAREGADDR (user_PMAREGADDR),

    .txpol (user_txpol),
    .txreset (user_txreset),
    .txscram64b66buse (user_txscram64b66buse),
    .txusrclk (user_txusrclk),
    .txusrclk2 (user_txusrclk2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.COMMA_10B_MASK = 10'b0011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "31_8";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

GT10_OC48_2 Verilog Code

GT10_OC48_2 GT10_OC48_2_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .PMARKLOCK (user_PMARKLOCK),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),

    .BREFCLKNIN (user_BREFCLKNIN),
    .BREFCLKPIN (user_BREFCLKPIN),
    .CHBONDI (user_CHBONDI),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .PMAINIT (user_PMAINIT),
    .PMAREGADDR (user_PMAREGADDR),

    .txpol (user_txpol),
    .txreset (user_txreset),
    .txscram64b66buse (user_txscram64b66buse),
    .txusrclk (user_txusrclk),
    .txusrclk2 (user_txusrclk2));
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGRW (user_PMAREGRW),
.PMAREGSTROBE (user_PMAREGSTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKKSEL (user_REFCLKKSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADEDETUSE (user_RXCOMMADEDETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARISMPHASE (user_TXCHARISMPHASE),
.TXCHARISMPHASE (user_TXCHARISMPHASE),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGARBOX64B66BUSE (user_TXGARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSREAM64B66BUSE (user_TXSREAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.COMMA_10B_MASK = 10'b0011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "31_16";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";
GT10_OC48_4 Code

GT10_OC48_4 GT10_OC48_4_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONO (user_CHBONO),
  .PMARXLOCK (user_PMARXLOCK),
  .RxBFST (user_RXBUFSTATUS),
  .RXCHARISO (user_RXCHARISOMMA),
  .RXCHARISK (user_RXCHARISOK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADE (user_RXCOMMADE),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECECL (user_RXRECECL),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXKERR (user_TXKERR),
  .TXN (user_TXN),
  .TXOUTCLK (user_TXOUTCLK),
  .TXP (user_TXP),
  .TXRUNDISP (user_TXRUNDISP),

  .BREFCLKNIN (user_BREFCLKNIN),
  .BREFCLKPIN (user_BREFCLKPIN),
  .CHBOND (user_CHBOND),
  .ENCHAN (user_ENCHAN),
  .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
  .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
  .LOOPBACK (user_LOOPBACK),
  .PMAININT (user_PMAININT),
  .PMAREADDR (user_PMAREADDR),
  .PMARDETA (user_PMARDETA),
  .PMAREGRW (user_PMAREGRW),
  .PMARESTROBE (user_PMARESTROBE),
  .PMARXLOCKSEL (user_PMARXLOCKSEL),
  .POWERDOWN (user_POWERDOWN),
  .REFCLK (user_REFC),
  .REFCLK2 (user_REFC),
  .REFCLKSEL (user_REFC),
  .RXBLOCSYNC64B66USE (user_RXBLOCSYNC64B66USE),
  .RXCOMMADEUSE (user_RXCOMMADEUSE),
  .RXDATAWIDTH (user_RXDATAWIDTH),
  .RXDECC (user_RXDEC64B66USE),
  .RXDEC8B10USE (user_RXDEC8B10USE),
  .RXDES (user_RXDESCRAM64B66USE),
  .RXIGNOR (user_RXIGNOR),
  .RXINTDATAW (user_RXINTDATAW),
  .RXN (user_RXN),
  .RXP (user_RXP),
  .RXPOLARITY (user_RXPOLARITY),
  .RXRESET (user_RXRESET),
  .RXSLIDE (user_RXSLIDE),
)
.RXUSRCLK (user_RXUSRCLK),
  .RXUSRCLK2 (user_RXUSRCLK2),
  .TXBYPASS8B10B (user_TXBYPASS8B10B),
  .TXCHARDISPMODE (user_TXCHARDISPMODE),
  .TXCHARDISPVAL (user_TXCHARDISPVAL),
  .TXCHARISK (user_TXCHARISK),
  .TXDATA (user_TXDATA),
  .TXDATAWIDTH (user_TXDATAWIDTH),
  .TXENC64B66BUSE (user_TXENC64B66BUSE),
  .TXENC8B10BUSE (user_TXENC8B10BUSE),
  .TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
  .TXINHIBIT (user_TXINHIBIT),
  .TXINTDATAWIDTH (user_TXINTDATAWIDTH),
  .TXPOLARITY (user_TXPOLARITY),
  .TXRESET (user_TXRESET),
  .TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
  .TXUSRCLK (user_TXUSRCLK),
  .TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.COMMA_10B_MASK = 10'b0011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "31_32";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

Commonly Used Constraints

None
GT10_OC192_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_OC192_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Xilinx protocol 10-gigabit transceiver supports 4 and 8-byte data paths. The letter \( n \) represents number of bytes of the data path. Valid values are 4 or 8.

You can also set attributes for the primitives. See the RocketIO X Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of \( n \). For a description of each of the ports, see the RocketIO X Transceiver User Guide.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDONE</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td>CHBOND(4.0)</td>
</tr>
<tr>
<td>CHBOND(4.0)</td>
<td>CHBOND(4.0)</td>
</tr>
<tr>
<td>ENCHANTSYNC</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>EMMCMAALIGN</td>
<td>PMAREG</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSTATUS(1:0)</td>
</tr>
<tr>
<td>PMANIT</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(3:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td>RXCHARISK(3:0)</td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>RXHARISK(3:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td>RXHARISK(3:0)</td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td>RXLOCKED</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>RXDATA(31:0)</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXDISPERR(3:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td>RXLOSSOSYSYN(1:0)</td>
</tr>
<tr>
<td>RXIGNOREBF</td>
<td>RXLOSTSYNC(1:0)</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RXNOTINTABLE(3:0)</td>
</tr>
<tr>
<td>RXN</td>
<td>RXN</td>
</tr>
<tr>
<td>RXP</td>
<td>RXP</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXCLEAR</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>RXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(3:0)</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>TXCHARISMTOME(3:0)</td>
<td>TXBUFFERR</td>
</tr>
<tr>
<td>TXCHARISHPMODE(3:0)</td>
<td>TXBUFFERR</td>
</tr>
<tr>
<td>TXCHARISHPVAL(3:0)</td>
<td>TXBUFFERR</td>
</tr>
<tr>
<td>TXCHARISH(3:0)</td>
<td>TXBUFFERR</td>
</tr>
<tr>
<td>TXDATA(31:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXGARLXXXX64B66BUSE</td>
<td>TXPO</td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td>TXPO</td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td>TXRUNDISP(3:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td>TXPOLARITY</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TXRESET</td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td>TXSCRAM64B66BUSE</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>Internal Signal</td>
<td>User Signal</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>BREFCLKNIN</td>
<td>CHBONDDONE</td>
</tr>
<tr>
<td>BREFCLKIN</td>
<td></td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
<td>CHBONDO(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSIZE(1:0)</td>
</tr>
<tr>
<td>PMAINIT</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(7:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td></td>
</tr>
<tr>
<td>PMAREGRW</td>
<td>RXCHARIS(7:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARKLOCKSEL(1:0)</td>
<td>RXCLKCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td></td>
</tr>
<tr>
<td>REFCLKBSSEL</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td>RXDATA(63:0)</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>RXDISPERR(7:0)</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXIGNOREBTF</td>
<td></td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RXNOTINTABLE(7:0)</td>
</tr>
<tr>
<td>RXIN</td>
<td></td>
</tr>
<tr>
<td>RXP</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>RXRESET</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXSLIDE</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td>RXRUNDISP(7:0)</td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td></td>
</tr>
<tr>
<td>TXBYPASS8B10B(7:0)</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>TXCHARDISPMODE(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARDISPMODVAL(7:0)</td>
<td>TXKERR(7:0)</td>
</tr>
<tr>
<td>TXCHARIS(7:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(63:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXENC64B66BUSE</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXENC8B10BUSE</td>
<td></td>
</tr>
<tr>
<td>TXGEARBOXES64B66BUSE</td>
<td>TXP</td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td>TXRUNDISP(7:0)</td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Instantiation Templates

GT10_OC192_4 VHDL Code

-- Component Attribute specification for GT10_OC192_4
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_OC192_4
-- synopsys translate_off
generic (  
  ALIGN_COMMA_WORD : integer := 1;
  COMMA_10B_MASK   : bit_vector := "0011111111";
  DEC_MCOMMA_DETECT: boolean := TRUE;
  DEC_PCOMMA_DETECT: boolean := TRUE;
  MCOMMA_10B_VALUE : bit_vector := "0010101010";
  MCOMMA_DETECT   : boolean := TRUE;
  PCOMMA_10B_VALUE : bit_vector := "0010101010";
  PCOMMA_DETECT   : boolean := TRUE;
  PMA_PWR_CNTRL   : bit_vector := "11111111";
  PMA_SPEED       : string := "15_32";
  RX_BUFFER_USE   : boolean := TRUE;
  RX_LOS_INVALID_INCR : integer := 1;
  RX_LOS_THRESHOLD: integer := 4;
  RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
  TX_BUFFER_USE   : boolean := TRUE);
-- synopsys translate_on
port (  
  BREFCLKNOUT     : out std_ulogic;
  BREFCLKPOUT     : out std_ulogic;
  CHBONDDONE      : out std_ulogic;
  CHBONDO         : out std_logic_vector(4 downto 0);
  PMARXLOCK       : out std_ulogic;
  RXBUFSTATUS     : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA   : out std_logic_vector(3 downto 0);
  RXCHARISK       : out std_logic_vector(3 downto 0);
  RXCLKCORCNT     : out std_logic_vector(2 downto 0);
  RXCOMMADET      : out std_logic;
  RXDATA          : out std_logic_vector(31 downto 0);
  RDXISPERR       : out std_logic_vector(3 downto 0);
  RXLOSSOFSYNC    : out std_logic_vector(1 downto 0);
  RXNOTINTABLE    : out std_logic_vector(3 downto 0);
  RXREALIGN       : out std_ulogic;
  RXRECLK         : out std_logic;
  RXRUNDISP       : out std_logic_vector(3 downto 0);
  TXBUFERR        : out std_ulogic;
  TXKERR          : out std_logic_vector(3 downto 0);
  TXN             : out std_ulogic;
  TXOUTCLK        : out std_ulogic;
  TXP             : out std_ulogic;
  TXRUNDISP       : out std_logic_vector(3 downto 0);
  BREFCLKNIN      : in std_ulogic;
)
end component;

-- Component Attribute specification for GT10_OC192_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_OC192_4 should be placed
-- in architecture after the begin keyword

GT10_OC192_4_INSTANCE_NAME : GT10_OC192_4
-- synopsys translate_off
generic map (  
  ALIGN_COMMA_WORD => integer_value,  
  COMMA_10B_MASK => bit_value,  
  DEC_MCOMMA_DETECT => boolean_value,  
  DEC_FCOMMA_DETECT => boolean_value,  
  MCOMMA_10B_VALUE => bit_value,  
  MCOMMA_DETECT => boolean_value,  
  FCOMMA_10B_VALUE => bit_value,  
  FCOMMA_DETECT => boolean_value,  
  PMA_PWR_CNTRL => bit_value,  
  PMA_SPEED => string_value,  
  RX_BUFFER_USE => boolean_value,  
  RX_LOS_INVALID_INCR => integer_value,  
  RX_LOS_THRESHOLD => integer_value,  
  RX_LOSS_OF_SYNC_FSM => boolean_value,  
  TX_BUFFER_USE => boolean_value)
-- synopsys translate_on
port map (  
  BREFCLKNOUT => user_BREFCLKNOUT,  
  BREFCLKPOUT => user_BREFCLKPOUT,  
  CHBONDDONE => user_CHBONDDONE,  
  CHBONO => user_CHBONO,  
  PMARXLOCK => user_PMARXLOCK,  
  RXBUFSTATUS => user_RXBUFSTATUS,  
  RXCHARISCOMMA => user_RXCHARISCOMMA,  
  RXCHARISK => user_RXCHARISK,  
  RXCLKCORCNT => user_RXCLKCORCNT,  
  RXCOMMADET => user_RXCOMMADET,  
  RXDATA => user_RXDATA,  
  RXDISPERR => user_RXDISPERR,  
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,  
  RXNOTINTABLE => user_RXNOTINTABLE,  
  RXREALIGN => user_RXREALIGN,  
  RXRECCCLK => user_RXRECCCLK,  
  RXRUNDISP => user_RXRUNDISP,  
  TXBUFERR => user_TXBUFERR,  
  TXKERR => user_TXKERR,  
  TXN => user_TXN,  
  TXOUTCLK => user_TXOUTCLK,  
  TXP => user_TXP,  
  TXRUNDISP => user_TXRUNDISP,  
  BREFCLKNIN => user_BREFCLKNIN,  
  BREFCLKPIN => user_BREFCLKPIN,  
  CHBONDI => user_CHBONDI,  
  ENCHANSYNC => user_ENCHANSYNC,  
  ENMCOMMAALIGN => user_ENMCOMMAALIGN,  
  ENPCOMMAALIGN => user_ENPCOMMAALIGN,  
  LOOPBACK => userLOOPBACK,  
  PMAINIT => user_PMAINIT,  
  PMAREGADDR => user_PMAREGADDR,  
  PMAREGDATAIN => user_PMAREGDATAIN,  
  PMAREGRW => user_PMAREGRW,  
  PMAREGSTROBE => user_PMAREGSTROBE,  
  PMARXLOCKSEL => user_PMARXLOCKSEL,  
  POWERDOWN => user_POWERDOWN,  
  REFCLKBSEL => user_REFCLKBSEL)
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETECTUSE => user_RXCOMMADETECTUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);

GT10_OC192_8 VHDL Code
-- Component Attribute specification for GT10_OC192_8
-- should be placed after architecture declaration but
-- before the begin keyword

cOMPONENT GT10_OC192_8
-- synopsys translate_off
generic (  
  ALIGN_COMMA_WORD : integer := 1;
  COMMA_10B_MASK : bit_vector := "0011111111";
  DEC_MCOMMA_DETECT : boolean := TRUE;
  DEC_PCOMMA_DETECT : boolean := TRUE;
  MCOMMA_10B_VALUE : bit_vector := "0010101010";
  MCOMMA_DETECT : boolean := TRUE;
  PCOMMA_10B_VALUE : bit_vector := "0010101010";
  PCOMMA_DETECT : boolean := TRUE;
  PMA_PWR_CNTRL : bit_vector := "11111111";
  PMA_SPEED : string := "15_64";
  RX_BUFFER_USE : boolean := TRUE;
  RX_LOS_INVALID_INCR : integer := 1;
  RX_LOS_THRESHOLD : integer := 4;
  RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
  TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port ( 
    BREFCLKNOUT : out std_ulogic;
    BREFCLKPOUT : out std_ulogic;
    CHBONDDONE  : out std_ulogic;
    CHBONDOT    : out std_logic_vector(4 downto 0);
    PMARXLOCK   : out std_ulogic;
    RXBUFSSTATUS: out std_logic_vector(1 downto 0);
    RXCHARISCOMMA: out std_logic_vector(7 downto 0);
    RXCHARISK   : out std_logic_vector(7 downto 0);
    RXCLKCORCNT : out std_logic_vector(2 downto 0);
    RXCOMMADET  : out std_ulogic;
    RXDATA      : out std_logic_vector(63 downto 0);
    RXDISPERR   : out std_logic_vector(7 downto 0);
    RXLOSSOFSYNC: out std_logic_vector(1 downto 0);
    RXNOTINTABLE: out std_logic_vector(7 downto 0);
    RXREALIGN   : out std_ulogic;
    RXRECCRCLK  : out std_ulogic;
    RXRUNDISP   : out std_logic_vector(7 downto 0);
    TXBUFErr    : out std_logic_vector(7 downto 0);
    TXKERR      : out std_logic_vector(7 downto 0);
    TXN         : out std_ulogic;
    TXOUTCLK    : out std_ulogic;
    TXP         : out std_ulogic;
    TXRUNDISP   : out std_logic_vector(7 downto 0);
    BREFCLKNIN : in std_ulogic;
    BREFCLKPIN : in std_ulogic;
    CHBONDI     : in std_logic_vector(4 downto 0);
    ENCHANSYNC  : in std_ulogic;
    ENPCOMMAALIGN: in std_ulogic;
    PMAREGADDR  : in std_logic_vector(5 downto 0);
    PMAREGDATAIN: in std_logic_vector(7 downto 0);
    PMAREGREGW  : in std_ulogic;
    PMAREGSTROBE: in std_ulogic;
    PMARXLOCKSEL: in std_logic_vector(1 downto 0);
    POWERDOWN   : in std_ulogic;
    REFCLKBSEL  : in std_ulogic;
    RXBLOCKSYNC64B66BUSE : in std_ulogic;
    RXCOMMADETTUSE : in std_ulogic;
    RXDATAWIDTH  : in std_logic_vector(1 downto 0);
    RXBypass8B10B : in std_logic_vector(7 downto 0);
    RXCHARDISPMODE: in std_logic_vector(7 downto 0);
    RXCHARDISPVAL: in std_logic_vector(7 downto 0);
    RXCHARISK   : in std_logic_vector(7 downto 0);)
end component;

-- Component Attribute specification for GT10_OC192_8
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_OC192_8 should be placed
-- in architecture after the begin keyword

GT10_OC192_8_INSTANCE_NAME : GT10_OC192_8
-- synopsys translate_off

generic map (
  ALIGN_COMMA_WORD          => integer_value,
  COMMA_10B_MASK            => bit_value,
  DEC_MCOMMA_DETECT         => boolean_value,
  DEC_PCOMMA_DETECT         => boolean_value,
  MCOMMA_10B_VALUE          => bit_value,
  MCOMMA_DETECT             => boolean_value,
  PCOMMA_10B_VALUE          => bit_value,
  PCOMMA_DETECT             => boolean_value,
  PMA_PWR_CNTRL             => bit_value,
  PMA_SPEED                 => string_value,
  RX_BUFFER_USE             => boolean_value,
  RX_LOS_INVALID_INCR       => integer_value,
  RX_LOS_THRESHOLD          => integer_value,
  RX_LOSS_OF_SYNC_FSM       => boolean_value,
  TX_BUFFER_USE             => boolean_value)
-- synopsys translate_on

port map (
  BREFCLKNOUT               => user_BREFCLKNOUT,
  BREFCLKPOUT               => user_BREFCLKPOUT,
  CHBONDDONE                => user_CHBONDDONE,
  CHBONO                     => user_CHBONO,
  PMARXLOCK                  => user_PMARXLOCK,
  RXBUFFSTATUS               => user_RXBUFFSTATUS,
  RXCHARISCOMMA             => user_RXCHARISCOMMA,
  RXCHARISK                  => user_RXCHARISK,
  RXCLKCORCNT                => user_RXCLKCORCNT,
  RXCOMMADET                 => user_RXCOMMADET,
  RXDATA                     => user_RXDATA,
  RXDISPERR                  => user_RXDISPERR,
  RXLOSSOFSYNC              => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXOUTCLK => user_TXOUTCLK,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLKNIN => user_BREFCLKNIN,
BREFCLKPIN => user_BREFCLKPIN,
CHBONDI => user_CHBONDI,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMAKLOCKSEL => user_PMAKLOCKSEL,
POWERDOWN => user_POWERDOWN,
REFCLKBSEL => user_REFCLKBSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATAWIDTH => user_RXDATAWIDTH,
RXDECS64B66BUSE => user_RXDECS64B66BUSE,
RXDECS810BUSE => user_RXDECS810BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXP => user_RXP,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC810BUSE => user_TXENC810BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);
Verilog Instantiation Templates

GT10_OC192_4 Verilog Code

```verilog
GT10_OC192_4 GT10_OC192_4_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONDO (user_CHBONDO),
    .PMARXLOCK (user_PMARXLOCK),
    .RXXBUFSTATUS (user_RXXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISSPERR (user_RXDISSPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECCLOCK (user_RXRECCLOCK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXKERR (user_TXKERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
    .TXRUNDISP (user_TXRUNDISP),
    .BREFCLKNIN (user_BREFCLKNIN),
    .BREFCLKPIN (user_BREFCLKPIN),
    .CHBONDI (user_CHBONDI),
    .ENCHANSYNC (user_ENCHANSYNC),
    .ENMCOMMAALIGN (user_ENMCOMMAALIGN),
    .ENPCOMMAALIGN (user_ENPCOMMAALIGN),
    .LOOPBACK (user_LOOPBACK),
    .PMAINIT (user_PMAINIT),
    .PMAREGADDR (user_PMAREGADDR),
    .PMAREGDATAIN (user_PMAREGDATAIN),
    .PMAREGRW (user_PMAREGRW),
    .PMAREGSTROBE (user_PMAREGSTROBE),
    .PMARXLOCKSEL (user_PMARXLOCKSEL),
    .POWERDOWN (user_POWERDOWN),
    .REFCLKBSEL (user_REFCLKBSEL),
    .RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
    .RXCOMMADETUSE (user_RXCOMMADETUSE),
    .RXDATAWIDTH (user_RXDATAWIDTH),
    .RXDEC64B66BUSE (user_RXDEC64B66BUSE),
    .RXDEC8B10BUSE (user_RXDEC8B10BUSE),
    .RXDECSCRAM64B66BUSE (user_RXDECSCRAM64B66BUSE),
    .RXIGNOREBTF (user_RXIGNOREBTF),
    .RXINTDATAWIDTH (user_RXINTDATAWIDTH),
    .RXN (user_RXN),
    .RXP (user_RXP),
    .RXPOLARITY (user_RXPOLARITY),
    .RXRESET (user_RXRESET),
    .RXSLIDE (user_RXSLIDE),
);```

.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.COMMA_10B_MASK = 10'b0011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "15_32";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

GT10_OC192_8 Verilog Code

GT10_OC192_8 GT10_OC192_8_name(
  .BREFCLKNOUT (user_BREFCLKNOUT),
  .BREFCLKPOUT (user_BREFCLKPOUT),
  .CHBONDDONE (user_CHBONDDONE),
  .CHBONO (user_CHBONO),
  .PMARKLOCK (user_PMARKLOCK),
  .RXBUFSTATUS (user_RXBUFSTATUS),
  .RXCHARISCOMMA (user_RXCHARISCOMMA),
  .RXCHARISK (user_RXCHARISK),
  .RXCLKCORCNT (user_RXCLKCORCNT),
  .RXCOMMADET (user_RXCOMMADET),
  .RXDATA (user_RXDATA),
  .RXDISPERR (user_RXDISPERR),
  .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
  .RXNOTINTABLE (user_RXNOTINTABLE),
  .RXREALIGN (user_RXREALIGN),
  .RXRECCCLK (user_RXRECCCLK),
  .RXRUNDISP (user_RXRUNDISP),
  .TXBUFERR (user_TXBUFERR),
  .TXERR (user_TXERR),
  .TXN (user_TXN),
defparam user_instance_name.ALIGN_COMMA_WORD = 1;
defparam user_instance_name.COMMA_10B_MASK = 10'b0011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0010101010;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "15_64";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

Commonly Used Constraints

None
GT10_PCI_EXPRESS_n

10-Gigabit Transceiver for High-Speed I/O

Architectures Supported

<table>
<thead>
<tr>
<th>GT10_PCI_EXPRESS_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

*Supported for Virtex-II Pro X but not for Virtex-II or Virtex-II Pro.

This Xilinx protocol 10-gigabit transceiver supports 1, 2 and 4-byte data paths. The letter n represents number of bytes of the data path. Valid values are 1, 2, or 4.

You can also set attributes for the primitives. See the RocketIO X Transceiver User Guide for a description of these attributes and their default attribute values.

The following figures list the input and output ports for all values of n. For a description of each of the ports, see the RocketIO X Transceiver User Guide.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>BREFCLKIN</td>
</tr>
<tr>
<td>CHBDONE</td>
<td>CHBDONE</td>
</tr>
<tr>
<td>PMAIN</td>
<td>PMAIN</td>
</tr>
<tr>
<td>PMARXLOCK</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>RXBUFSIZE</td>
<td>RXBUFSIZE</td>
</tr>
<tr>
<td>RXCOMMADET</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>RXDATA(7:0)</td>
<td>RXDATA(7:0)</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXDATAWIDTH(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td>RXDEC64B66BUSE</td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXDEC8B10BUSE</td>
</tr>
<tr>
<td>RXRECICLK</td>
<td>RXRECICLK</td>
</tr>
<tr>
<td>RXPOWERT</td>
<td>RXPOWERT</td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXPOLARITY</td>
</tr>
<tr>
<td>RXRUNDISP</td>
<td>RXRUNDISP(0:0)</td>
</tr>
<tr>
<td>RXBUFERR</td>
<td>RXBUFERR</td>
</tr>
<tr>
<td>TXBUFERR</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>TXGOODF</td>
<td>TXGOODF</td>
</tr>
<tr>
<td>TXIERR</td>
<td>TXIERR(0:0)</td>
</tr>
<tr>
<td>TXPOWERT</td>
<td>TXPOWERT</td>
</tr>
<tr>
<td>TXPOWERT</td>
<td>TXPOWERT</td>
</tr>
<tr>
<td>TXRESET</td>
<td>TXRESET</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td>TXUSRCLK</td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td>TXUSRCLK2</td>
</tr>
<tr>
<td>TXUOUTCLK</td>
<td>TXUOUTCLK</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>TXRUNDISP(0:0)</td>
</tr>
<tr>
<td>TXRUNDISP</td>
<td>TXRUNDISP(0:0)</td>
</tr>
</tbody>
</table>
**GT10_PCI_EXPRESS_2**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREFCLKIN</td>
<td>CHBONDONE</td>
</tr>
<tr>
<td>BREFCLKPIN</td>
<td></td>
</tr>
<tr>
<td>CHBOND(4:0)</td>
<td>CHBOND(4:0)</td>
</tr>
<tr>
<td>ENCHANSYNC</td>
<td></td>
</tr>
<tr>
<td>ENMCOMMAALIGN</td>
<td>PMARXLOCK</td>
</tr>
<tr>
<td>ENPCOMMAALIGN</td>
<td></td>
</tr>
<tr>
<td>LOOPBACK(1:0)</td>
<td>RXBUFSIZE(1:0)</td>
</tr>
<tr>
<td>PMAIN</td>
<td></td>
</tr>
<tr>
<td>PMAREGADDR(5:0)</td>
<td>RXCHARISCOMMA(1:0)</td>
</tr>
<tr>
<td>PMAREGDATAIN(7:0)</td>
<td></td>
</tr>
<tr>
<td>PMAREGRNW</td>
<td>RXCHARISK(1:0)</td>
</tr>
<tr>
<td>PMAREGSTROBE</td>
<td></td>
</tr>
<tr>
<td>PMARXLOCKSEL(1:0)</td>
<td>RXCLCORCNT(2:0)</td>
</tr>
<tr>
<td>POWERDOWN</td>
<td></td>
</tr>
<tr>
<td>REFCLK</td>
<td>RXCOMMADET</td>
</tr>
<tr>
<td>REFCLK2</td>
<td></td>
</tr>
<tr>
<td>REFCLKBSEL</td>
<td>RXDATA(15:0)</td>
</tr>
<tr>
<td>REFCLKSEL</td>
<td></td>
</tr>
<tr>
<td>RXBLOCKSYNC64B66BUSE</td>
<td>RXDISERR(1:0)</td>
</tr>
<tr>
<td>RXCOMMADETUSE</td>
<td></td>
</tr>
<tr>
<td>RXDATAWIDTH(1:0)</td>
<td>RXLOSSOFSYNC(1:0)</td>
</tr>
<tr>
<td>RXDEC64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXDEC8B10BUSE</td>
<td>RXNINTABLE(1:0)</td>
</tr>
<tr>
<td>RXDESCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>RXIGNOREBITF</td>
<td>RXREALIGN</td>
</tr>
<tr>
<td>RXINTDATAWIDTH(1:0)</td>
<td>RXRECLK</td>
</tr>
<tr>
<td>RXN</td>
<td></td>
</tr>
<tr>
<td>RXP</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY</td>
<td>RXRUNDISP(1:0)</td>
</tr>
<tr>
<td>RXRESET</td>
<td></td>
</tr>
<tr>
<td>RXSLIDE</td>
<td>TXBUFERR</td>
</tr>
<tr>
<td>RXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>RXUSRCLK2</td>
<td>TXERR(1:0)</td>
</tr>
<tr>
<td>TXBYPASS8B10B(1:0)</td>
<td>TXN</td>
</tr>
<tr>
<td>TXCHARISDMODE(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXCHARISPMODEN(1:0)</td>
<td>TXOUTCLK</td>
</tr>
<tr>
<td>TXCHARISK(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATA(15:0)</td>
<td></td>
</tr>
<tr>
<td>TXDATAWIDTH(1:0)</td>
<td>TXP</td>
</tr>
<tr>
<td>TXENCODE64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXENCODEB10BUSE</td>
<td>TXRUNDISP(1:0)</td>
</tr>
<tr>
<td>TXGERBOX64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXINHIBIT</td>
<td></td>
</tr>
<tr>
<td>TXINTDATAWIDTH(1:0)</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY</td>
<td></td>
</tr>
<tr>
<td>TXRESET</td>
<td></td>
</tr>
<tr>
<td>TXSCRAM64B66BUSE</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK</td>
<td></td>
</tr>
<tr>
<td>TXUSRCLK2</td>
<td></td>
</tr>
</tbody>
</table>
Usage

This design element is instantiated rather than inferred in the design code.

VHDL Start

VHDL Instantiation Templates

GT10_PCI_EXPRESS_1 Code

-- Component Attribute specification for GT10_PCI_EXPRESS_1
-- should be placed after architecture declaration but
-- before the begin keyword

component GT10_PCI_EXPRESS_1
   -- synopsys translate_off
generic (  
   ALIGN_COMMA_WORD : integer := 2;
   CHAN_BOND_LIMIT  : integer := 16;
   CHAN_BOND_MODE   : string := "OFF";
   CHAN_BOND_ONE_SHOT : boolean := FALSE;
   CHAN_BOND_SEQ_1_1 : bit_vector := "00110111100";
   CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_1_3 : bit_vector := "0001001010";
   CHAN_BOND_SEQ_1_4 : bit_vector := "0001001010";
   CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
   CHAN_BOND_SEQ_2_1 : bit_vector := "00110111100";
   CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
   CHAN_BOND_SEQ_2_3 : bit_vector := "0001001010";
   CHAN_BOND_SEQ_2_4 : bit_vector := "0001001010";
   CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
   CHAN_BOND_SEQ_2_USE : boolean := FALSE;
   CHAN_BOND_SEQ_LEN : integer := 2;
   CLK_COR_8B10B_DE : boolean := FALSE;
   CLK_COR_MAX_LAT : integer := 36;
   CLK_COR_MIN_LAT : integer := 28;
   CLK_COR_SEQ_1_1 : bit_vector := "00100011100";
   CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
   CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
   CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
   CLK_COR_SEQ_1_MASK : bit_vector := "0000";
   CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_3 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_4 : bit_vector := "00000000000";
   CLK_COR_SEQ_2_MASK : bit_vector := "0000";
   CLK_COR_SEQ_2_USE : boolean := FALSE;
   CLK_COR_SEQ_DROP : boolean := FALSE;
   CLK_COR_SEQ_LEN : integer := 2;
   CLK_CORRECT_USE : boolean := TRUE;
   COMMA_10B_MASK : bit_vector := "0001111111";
   DEC_MCOMMA_DETECT : boolean := TRUE;
   DEC_PCOMMA_DETECT : boolean := TRUE;
   DEC_VALID_COMMA_ONLY : boolean := TRUE;
   MCOMMA_10B_VALUE : bit_vector := "1010000011";
   MCOMMA_DETECT : boolean := TRUE;
   PCOMMA_10B_VALUE : bit_vector := "0101111100";
   PCOMMA_DETECT : boolean := TRUE;  
)
-- synopsys translate_on

port (  
  BREFCLKNOUT               : out std_ulogic;
  BREFCLKPIN                : in std_ulogic;
  CHBONDI                   : in std_logic_vector(4 downto 0);
  CHBONDONE                : out std_logic;
  CHBONDO                  : out std_logic_vector(4 downto 0);
  PMAREGADDR                : in std_logic_vector(5 downto 0);
  PMAREGDATAIN              : in std_logic_vector(7 downto 0);
  PMAREGINIT                : in std_ulogic;
  PMAREGSTROBE              : in std_logic;
  PMARXLOCK                 : out std_ulogic;
  PMARXLOCKSEL              : in std_logic_vector(1 downto 0);
  PMARXLOCKSEL              : in std_logic;
  RXBLOCKSYNC64B66BUSE      : in std_logic;
  RXBUFFER_USE             : boolean := TRUE;
  RXBUFSTATUS               : out std_logic_vector(1 downto 0);
  RXBUFSTUS                : out std_logic_vector(1 downto 0);
  RXCHARISK                 : out std_logic_vector(0 downto 0);
  RXCHARISCOMMA             : out std_logic_vector(0 downto 0);
  RXCOMMADET                : out std_ulogic;
  RXCOMMADET                : out std_logic;
  RXDATA                    : out std_logic_vector(7 downto 0);
  RXDATAWIDTH               : in std_logic_vector(1 downto 0);
  RXDISPERR                 : out std_logic_vector(0 downto 0);
  RXDISPERR                : out std_logic_vector(0 downto 0);
  RXLOSSOFSYNC              : out std_logic_vector(1 downto 0);
  RXLOSSOFSYNC              : out std_logic_vector(1 downto 0);
  RXNOTINTABLE              : out std_logic_vector(0 downto 0);
  RXNOTINTABLE              : out std_logic_vector(0 downto 0);
  RXREALIGN                 : out std_ulogic;
  RXREALIGN                 : out std_logic;
  RXRUNDISP                 : out std_logic_vector(0 downto 0);
  RXRUNDISP                 : out std_logic_vector(0 downto 0);
  TXBUFERR                  : out std_ulogic;
  TXBUFERR                  : out std_logic;
  TXKERR                    : out std_logic_vector(0 downto 0);
  TXKERR                    : out std_logic_vector(0 downto 0);
  TXN                       : out std_ulogic;
  TXN                       : out std_logic;
  TXOUTCLK                  : out std_ulogic;
  TXOUTCLK                  : out std_logic;
  TXP                       : out std_ulogic;
  TXP                       : out std_logic;
  RXDESCRIPTION64B66BUSE    : in std_logic;
  RXDESCRIPTION64B66BUSE    : in std_logic;
  RXDESCRIPTION8B10BUSE     : in std_logic;
  RXDESCRIPTION8B10BUSE     : in std_logic;
  RXDESCRIPTION64B66BUSE    : in std_logic;
  RXDESCRIPTION64B66BUSE    : in std_logic;
  RXDESCRIPTION64B66BUSE    : in std_logic;
  RXDESCRIPTION64B66BUSE    : in std_logic;
  RXIGNOREBTF               : in std_logic;
  RXIGNOREBTF               : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYNC64B66BUSE      : in std_logic;
  RXMERGESYN

PMA_PWR_CNTRL             : bit_vector := "11111111";
PMA_SPEED                  : string := "28_10";
RX_BUFFER_USE             : boolean := TRUE;
RX_LOS_INVALID_INCR        : integer := 1;
RX_LOS_THRESHOLD           : integer := 4;
RX_LOSS_OF_SYNC_FSM        : boolean := TRUE;
TX_BUFFER_USE             : boolean := TRUE);
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXSLIDE : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(0 downto 0);
TXCHARDISPMODE : in std_logic_vector(0 downto 0);
TXCHARDISPVAL : in std_logic_vector(0 downto 0);
TXCHARISK : in std_logic_vector(0 downto 0);
TXDATA : in std_logic_vector(7 downto 0);
TXDATAWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_ulogic;
TXENC8B10BUSE : in std_ulogic;
TXGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXSCRAM64B66BUSE : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

drop component;

-- Component Attribute specification for GT10_PCI_EXPRESS_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_PCI_EXPRESS_1 should be placed
-- in architecture after the begin keyword

GT10_PCI_EXPRESS_1_INSTANCE_NAME : GT10_PCI_EXPRESS_1

-- synopsys translate_off

generic map (
ALIGN_COMMA_WORD => integer_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CLK_COR_8B10B_DE => boolean_value,
CLK_COR_MAX_LAT => integer_value,
CLK_COR_MIN_LAT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
COMMA_10B_MASK => bit_value,
DEC_MCOMMA_DETECT => boolean_value,
DEC_PCOMMA_DETECT => boolean_value,
DEC_VALID_COMMA_ONLY => boolean_value,
MCOMMA_10B_VALUE => bit_value,
MCOMMA_DETECT => boolean_value,
MCOMMA_DETECT => boolean_value,
PMA_PWR_CNTRL => bit_value,
PMA_SPEED => string_value,
RX_BUFFER_USE => boolean_value,
RX_LOS_INVALID_INCR => integer_value,
RX_LOS_THRESHOLD => integer_value,
RX_LOSS_OF_SYNC_FSM => boolean_value,
RX_OFFSET => integer_value,
RX_REALIGN => boolean_value,
RXRECCLK => user_RXRECCLK,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
RXDISERR => user_RXDISERR,
RXLOSSOSYSYNC => user_RXLOSSOSYSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECCLK => user_RXRECCLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFFERR => user_TXBUFFERR,
TXERR => user_TXERR,
TXN => user_TXN,
TXOUTCLK => user_TXOUTCLK,
TXP => user_TXP,
TXRUNDISP => user_TXRUNDISP,
BREFCLKIN => user_BREFCLKIN,
BREFCLKP => user_BREFCLKP,
CHBONDONE => user_CHBONDONE,
CHBOND => user_CHBOND,
PMARXLOCK => user_PMARXLOCK,
RXBUFSTATUS => user_RXBUFSTATUS,
RXCHARISCOMMA => user_RXCHARISCOMMA,
RXCHARISK => user_RXCHARISK,
RXCLKCORCNT => user_RXCLKCORCNT,
RXCOMMADET => user_RXCOMMADET,
RXDATA => user_RXDATA,
--- Component Attribute specification for GT10_PCI_EXPRESS_2
--- should be placed after architecture declaration but
--- before the begin keyword

component GT10_PCI_EXPRESS_2
-- synopsys translate_off
generic (
  ALIGN_COMMA_WORD : integer := 2;
  CHAN_BOND_LIMIT : integer := 16;
);
CHAN_BOND_MODE : string := "OFF";
CHAN_BOND_ONE_SHOT : boolean := FALSE;
CHAN_BOND_SEQ_1_1 : bit_vector := "001101111000";
CHAN_BOND_SEQ_1_2 : bit_vector := "000000000000";
CHAN_BOND_SEQ_1_3 : bit_vector := "000010010100";
CHAN_BOND_SEQ_1_4 : bit_vector := "000010010100";
CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
CHAN_BOND_SEQ_2_1 : bit_vector := "001101111000";
CHAN_BOND_SEQ_2_2 : bit_vector := "000000000000";
CHAN_BOND_SEQ_2_3 : bit_vector := "000010010100";
CHAN_BOND_SEQ_2_4 : bit_vector := "000010010100";
CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
CHAN_BOND_SEQ_2_USE : boolean := FALSE;
CHAN_BOND_SEQ_LEN : integer := 2;
CLK_COR_8B10B_DE : boolean := FALSE;
CLK_COR_MAX_LAT : integer := 36;
CLK_COR_MIN_LAT : integer := 28;
CLK_COR_SEQ_1_1 : bit_vector := "001000111000";
CLK_COR_SEQ_1_2 : bit_vector := "000000000000";
CLK_COR_SEQ_1_3 : bit_vector := "000000000000";
CLK_COR_SEQ_1_4 : bit_vector := "000000000000";
CLK_COR_SEQ_1_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_1 : bit_vector := "000000000000";
CLK_COR_SEQ_2_2 : bit_vector := "000000000000";
CLK_COR_SEQ_2_3 : bit_vector := "000000000000";
CLK_COR_SEQ_2_4 : bit_vector := "000000000000";
CLK_COR_SEQ_2_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_USE : boolean := FALSE;
CLK_COR_SEQ_DROP : boolean := FALSE;
CLK_COR_SEQ_LEN : integer := 2;
COMMA_10B_MASK : bit_vector := "000111111111";
DEC_MCOMMA_DETECT : boolean := TRUE;
DEC_PCOMMA_DETECT : boolean := TRUE;
DEC_VALID_COMMA_ONLY : boolean := TRUE;
MCOMMA_10B_VALUE : bit_vector := "1010000011";
MCOMMA_DETECT : boolean := TRUE;
PCOMMA_10B_VALUE : bit_vector := "0101111100";
PCOMMA_DETECT : boolean := TRUE;
PMA_PWR_CNTRL : bit_vector := "11111111";
PMA_SPEED : string := "28_20";
RX_BUFFER_USE : boolean := TRUE;
RX_LOS_INVALID_INCR : integer := 1;
RX_LOS_THRESHOLD : integer := 4;
RX_LOSS_OF_SYNC_FSM : boolean := TRUE;
TX_BUFFER_USE : boolean := TRUE);

-- synopsys translate_on
port (
  BREFCLKNOUT : out std_ulogic;
  BREFCLKPOUT : out std_ulogic;
  CHBONDDONE : out std_ulogic;
  CHBONDO : out std_logic_vector(4 downto 0);
  PMARXLOCK : out std_ulogic;
  RXBUFSTATUS : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA : out std_logic_vector(1 downto 0);
  RXCHARISK : out std_logic_vector(1 downto 0);
  RXCLKCORCNT : out std_logic_vector(2 downto 0);
RXCOMMADET : out std_ulogic;
RXDATA : out std_logic_vector(15 downto 0);
RXDISPERR : out std_logic_vector(1 downto 0);
RXLOSSOFSYNC : out std_logic_vector(1 downto 0);
RXNOTINTABLE : out std_logic_vector(1 downto 0);
RXREALIGN : out std_ulogic;
RXRECCCLK : out std_ulogic;
RXRUNDISP : out std_logic_vector(1 downto 0);
TXBUFERR : out std_ulogic;
TXERR : out std_logic_vector(1 downto 0);
TXN : out std_ulogic;
TXOUTCLK : out std_ulogic;
TXP : out std_ulogic;
TXRUNDISP : out std_logic_vector(1 downto 0);
BREFCLKNIN : in std_ulogic;
BREFCLKPIN : in std_ulogic;
CHBONDI : in std_logic_vector(4 downto 0);
ENCHANSYNC : in std_ulogic;
ENMCOMMAALIGN : in std_ulogic;
ENPCOMMAALIGN : in std_ulogic;
LOOPBACK : in std_logic_vector(1 downto 0);
PMAINIT : in std_ulogic;
Pmaregaddr : in std_logic_vector(5 downto 0);
Pmaregdatain : in std_logic_vector(7 downto 0);
Pmaregrw : in std_ulogic;
Pmaregstrobe : in std_ulogic;
Pmarxlocksel : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKBSEL : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXBLOCKSYNC64B66BUSE : in std_ulogic;
RXCOMMADETUSE : in std_ulogic;
RXDATAWIDTH : in std_logic_vector(1 downto 0);
RXDEC64B66BUSE : in std_ulogic;
RXDEC8B10BUSE : in std_ulogic;
RXDESCRAM64B66BUSE : in std_ulogic;
RXIGNOREBTF : in std_ulogic;
RXINTDATAWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXSLIDE : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(1 downto 0);
TXCHARDISPMODE : in std_logic_vector(1 downto 0);
TXCHARDISPVAL : in std_logic_vector(1 downto 0);
TXCHARISK : in std_logic_vector(1 downto 0);
TXDATA : in std_logic_vector(15 downto 0);
TXDATAWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_ulogic;
TXENC8B10BUSE : in std_ulogic;
TXGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATAWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXSCRAM64B66BUSE : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

port map (
  BREFCLKNOUT => user_BREFCLKNOUT,
  BREFCLKPOUT => user_BREFCLKPOUT,
  CHBONDDONE => user_CHBONDDONE,
  CHBOND0 => user_CHBOND0,
  PMARXLOCK => user_PMARXLOCK,
  RXBUFSTATUS => user_RXBUFSTATUS,
  RXCHARISCOMMA => user_RXCHARISCOMMA,
  RXCHARISK => user_RXCHARISK,
  RXCLKCORCNT => user_RXCLKCORCNT,
  RXCOMMADET => user_RXCOMMADET,
  RXDATA => user_RXDATA,
  RXDISPERR => user_RXDISPERR,
  RXLOSSOFSYNC => user_RXLOSSOFSYNC,
  RXNOTINTABLE => user_RXNOTINTABLE,
  RXREALIGN => user_RXREALIGN,
  RXRECCCLK => user_RXRECCCLK,
  RXRUNDISP => user_RXRUNDISP,
  TXBUFFERR => user_TXBUFFERR,
  TXKERR => user_TXKERR,
  TXN => user_TXN,
  TXOUTCLK => user_TXOUTCLK,
  TXP => user_TXP,
  TXRUNDISP => user_TXRUNDISP,
  BREFCLKNIN => user_BREFCLKNIN,
  BREFCLKPIN => user_BREFCLKPIN,
  CHBONDI => user_CHBONDI,
  ENCHANSYNC => user_ENCHANSYNC,
  ENMICOMMAALIGN => user_ENMICOMMAALIGN,
  ENPCOMMAALIGN => user_ENPCOMMAALIGN,
  LOOPBACK => user_LOOPBACK,
  PMAINIT => user_PMAINIT,
  PMAREGADDR => user_PMAREGADDR,
  PMAREGDATAIN => user_PMAREGDATAIN,
  PMAREGRW => user_PMAREGRW,
  PMAREGSTROBE => user_PMAREGSTROBE,
  PMARXLOCKSEL => user_PMARXLOCKSEL,
  POWERDOWN => user_POWERDOWN,
  REFCLK => user_REFCLK,
  REFCLK2 => user_REFCLK2,
  REFCLKBSSEL => user_REFCLKBSSEL,
  REFCLKSEL => user_REFCLKSEL,
  RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
  RXCOMMADETAUSE => user_RXCOMMADETAUSE,
  RXDATAWIDTH => user_RXDATAWIDTH,
  RXDEC64B66BUSE => user_RXDEC64B66BUSE,
  RXDEC8B10BUSE => user_RXDEC8B10BUSE,
  RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
  RXIGNOREBF => user_RXIGNOREBF,
  RXINTDATAWIDTH => user_RXINTDATAWIDTH,
  RXN => user_RXN,
  RXP => user_RXP,
GT10_PCI_EXPRESS_4 Code

-- Component Attribute specification for GT10_PCI_EXPRESS_4
-- should be placed after architecture declaration but
-- before the begin keyword

compoment GT10_PCI_EXPRESS_4
-- synopsys translate_off
generic ( ALIGN_COMMA_WORD : integer := 2;
CHAN_BOND_LIMIT : integer := 16;
CHAN_BOND_MODE : string := "OFF";
CHAN_BOND_ONE_SHOT : boolean := FALSE;
CHAN_BOND_SEQ_1_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_1_2 : bit_vector := "00000000000";
CHAN_BOND_SEQ_1_3 : bit_vector := "00001001010";
CHAN_BOND_SEQ_1_4 : bit_vector := "00001001010";
CHAN_BOND_SEQ_1_MASK : bit_vector := "0000";
CHAN_BOND_SEQ_2_1 : bit_vector := "00110111100";
CHAN_BOND_SEQ_2_2 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_3 : bit_vector := "00000000000";
CHAN_BOND_SEQ_2_4 : bit_vector := "00001000101";
CHAN_BOND_SEQ_2_MASK : bit_vector := "0000";
CHAN_BOND_SEQ_2_USE : boolean := FALSE;
CHAN_BOND_SEQ_LEN : integer := 2;
CLK_COR_8B10B_DE : boolean := FALSE;
CLK_COR_MAX_LAT : integer := 36;
CLK_COR_MIN_LAT : integer := 28;
CLK_COR_SEQ_1_1 : bit_vector := "00100111100";
CLK_COR_SEQ_1_2 : bit_vector := "00000000000";
CLK_COR_SEQ_1_3 : bit_vector := "00000000000";
CLK_COR_SEQ_1_4 : bit_vector := "00000000000";
CLK_COR_SEQ_1_MASK : bit_vector := "0000";
CLK_COR_SEQ_2_1 : bit_vector := "00000000000";
CLK_COR_SEQ_2_2 : bit_vector := "00000000000";
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARISPVAL => user_TXCHARISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATAWIDTH => user_TXDATAWIDTH,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXGEARBOX64B66BUSE => user_TXGEARBOX64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
TXPOLARITY => user_TXPOLARITY,
TXRESET => user_TXRESET,
TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
TXUSRCLK => user_TXUSRCLK,
TXUSRCLK2 => user_TXUSRCLK2);
CLK_COR_SEQ_2_3          : bit_vector := "000000000000";
CLK_COR_SEQ_2_4          : bit_vector := "000000000000";
CLK_COR_SEQ_2_MASK      : bit_vector := "0000";
CLK_COR_SEQ_2_USE       : boolean := FALSE;
CLK_COR_SEQ_DROP        : boolean := FALSE;
CLK_COR_SEQ_LEN         : integer := 2;
CLK_CORRECT_USE         : boolean := TRUE;
COMMA_10B_MASK          : bit_vector := "0001111111";
DEC_MCOMMA_DETECT       : boolean := TRUE;
DEC_PCOMMA_DETECT       : boolean := TRUE;
DEC_VALID_Comma_ONLY    : boolean := TRUE;
MCOMMA_10B_VALUE        : bit_vector := "101000001111";
MCOMMA_DETECT           : boolean := TRUE;
PCOMMA_10B_VALUE        : bit_vector := "0101111100";
PCOMMA_DETECT           : boolean := TRUE;
PMA_PWR_CNTRL           : bit_vector := "1111111111";
PMA_SPEED               : string := "28_40";
RX_BUFFER_USE           : boolean := TRUE;
RX_LOS_INVALID_INCR     : integer := 1;
RX_LOS_THRESHOLD        : integer := 4;
RX_LOSS_OF_SYNC_FSM     : boolean := TRUE;
TX_BUFFER_USE           : boolean := TRUE);

-- synopsys translate_on
port (
  BREFCLKNOUT       : out std_ulogic;
  BREFCLKPOUT       : out std_ulogic;
  CHBONDDONE        : out std_ulogic;
  CHBONDO           : out std_logic_vector(4 downto 0);
  PMARXLOCK         : out std_ulogic;
  RXBUFSTATUS       : out std_logic_vector(1 downto 0);
  RXCHARISCOMMA     : out std_logic_vector(3 downto 0);
  RXCHARISK         : out std_logic_vector(3 downto 0);
  RXCLKCORCNT       : out std_logic_vector(2 downto 0);
  RXCOMMADET        : out std_ulogic;
  RXDATA            : out std_logic_vector(31 downto 0);
  RXDISPERR         : out std_logic_vector(3 downto 0);
  RXLOSSOFSYNC      : out std_logic_vector(3 downto 0);
  RXNOTINTABLE       : out std_logic_vector(3 downto 0);
  RXREALIGN         : out std_ulogic;
  RXRECLK           : out std_ulogic;
  RXRUNDISP         : out std_logic_vector(3 downto 0);
  TBUFFERR           : out std_ulogic;
  TXKERR            : out std_logic_vector(3 downto 0);
  TXN               : out std_ulogic;
  TXOUTCLK          : out std_ulogic;
  TXP               : out std_ulogic;
  TXRUNDISP         : out std_logic_vector(3 downto 0);
  BREFCLKNIN        : in std_ulogic;
  BREFCLKPIN        : in std_ulogic;
  CHBONDI           : in std_logic_vector(4 downto 0);
  ENCHANSSYNC       : in std_ulogic;
  ENMCOMMAALIGN     : in std_ulogic;
  ENFCOMMAALIGN     : in std_ulogic;
  LOOPBACK          : in std_logic_vector(1 downto 0);
  PMAINIT           : in std_ulogic;
  PMAREGADDR        : in std_logic_vector(5 downto 0);
  PMAREGDATAIN      : in std_logic_vector(7 downto 0);
PMAREGRW : in std_ulogic;
PMAREGSTROBE : in std_ulogic;
PMAXLOCKSEL : in std_logic_vector(1 downto 0);
POWERDOWN : in std_ulogic;
REFCLK : in std_ulogic;
REFCLK2 : in std_ulogic;
REFCLKBSEL : in std_ulogic;
REFCLKSEL : in std_ulogic;
RXBLOCKSYNC64B66BUSE : in std_ulogic;
RXCOMMADETUSE : in std_ulogic;
RXDATWIDTH : in std_logic_vector(1 downto 0);
RXDEC64B66BUSE : in std_ulogic;
RXDEC8B10BUSE : in std_ulogic;
RXDESCRAM64B66BUSE : in std_ulogic;
RXIGNOREBTF : in std_ulogic;
RXINTDATWIDTH : in std_logic_vector(1 downto 0);
RXN : in std_ulogic;
RXP : in std_ulogic;
RXPOLARITY : in std_ulogic;
RXRESET : in std_ulogic;
RXSLIDE : in std_ulogic;
RXUSRCLK : in std_ulogic;
RXUSRCLK2 : in std_ulogic;
TXBYPASS8B10B : in std_logic_vector(3 downto 0);
TXCHARDISPMODE : in std_logic_vector(3 downto 0);
TXCHARDISPVAL : in std_logic_vector(3 downto 0);
TXCHARISK : in std_logic_vector(3 downto 0);
TXDATA : in std_logic_vector(31 downto 0);
TXDATWIDTH : in std_logic_vector(1 downto 0);
TXENC64B66BUSE : in std_ulogic;
TXENC8B10BUSE : in std_ulogic;
TXGEARBOX64B66BUSE : in std_ulogic;
TXINHIBIT : in std_ulogic;
TXINTDATWIDTH : in std_logic_vector(1 downto 0);
TXPOLARITY : in std_ulogic;
TXRESET : in std_ulogic;
TXSCRAM64B66BUSE : in std_ulogic;
TXUSRCLK : in std_ulogic;
TXUSRCLK2 : in std_ulogic);

end component;

-- Component Attribute specification for GT10_PCI_EXPRESS_4
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter constraints here

-- Component Instantiation for GT10_PCI_EXPRESS_4 should be placed
-- in architecture after the begin keyword

GT10_PCI_EXPRESS_4_INSTANCE_NAME : GT10_PCI_EXPRESS_4

-- synopsys translate_off
generic map (  
ALIGN_COMMA_WORD => integer_value,
CHAN_BOND_LIMIT => integer_value,
CHAN_BOND_MODE => string_value,
CHAN_BOND_ONE_SHOT => boolean_value,
CHAN_BOND_SEQ_1_1 => bit_value,
CHAN_BOND_SEQ_1_2 => bit_value,
CHAN_BOND_SEQ_1_3 => bit_value,
CHAN_BOND_SEQ_1_4 => bit_value,
CHAN_BOND_SEQ_1_MASK => bit_value,
CHAN_BOND_SEQ_2_1 => bit_value,
CHAN_BOND_SEQ_2_2 => bit_value,
CHAN_BOND_SEQ_2_3 => bit_value,
CHAN_BOND_SEQ_2_4 => bit_value,
CHAN_BOND_SEQ_2_MASK => bit_value,
CHAN_BOND_SEQ_2_USE => boolean_value,
CHAN_BOND_SEQ_LEN => integer_value,
CLK_COR_8B10B_DE => boolean_value,
CLK_COR_MAX_LAT => integer_value,
CLK_COR_MIN_LAT => integer_value,
CLK_COR_SEQ_1_1 => bit_value,
CLK_COR_SEQ_1_2 => bit_value,
CLK_COR_SEQ_1_3 => bit_value,
CLK_COR_SEQ_1_4 => bit_value,
CLK_COR_SEQ_1_MASK => bit_value,
CLK_COR_SEQ_2_1 => bit_value,
CLK_COR_SEQ_2_2 => bit_value,
CLK_COR_SEQ_2_3 => bit_value,
CLK_COR_SEQ_2_4 => bit_value,
CLK_COR_SEQ_2_MASK => bit_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
CLK_COR_SEQ_2_USE => boolean_value,
RXCOMMADET => user_RXCOMMADET,
RXDATA => user_RXDATA,
RXDISPERR => user_RXDISPERR,
RXLOSSOFSYNC => user_RXLOSSOFSYNC,
RXNOTINTABLE => user_RXNOTINTABLE,
RXREALIGN => user_RXREALIGN,
RXRECLK => user_RXRECLK,
RXRUNDISP => user_RXRUNDISP,
TXBUFERR => user_TXBUFERR,
TXKERR => user_TXKERR,
TXN => user_TXN,
TXOUTCLK => user_TXOUTCLK,
TXF => user_TXF,
TXRUNDISP => user_TXRUNDISP,
BREFCLKNIN => user_BREFCLKNIN,
BREFCLKPIN => user_BREFCLKPIN,
CHBONDI => user_CHBONDI,
ENCHANSYNC => user_ENCHANSYNC,
ENMCOMMAALIGN => user_ENMCOMMAALIGN,
ENPCOMMAALIGN => user_ENPCOMMAALIGN,
LOOPBACK => user_LOOPBACK,
PMAINIT => user_PMAINIT,
PMAREGADDR => user_PMAREGADDR,
PMAREGDATAIN => user_PMAREGDATAIN,
PMAREGRW => user_PMAREGRW,
PMAREGSTROBE => user_PMAREGSTROBE,
PMArxLOCKSEL => user_PMAREGSTROBE,
POWERDOWN => user_POWERDOWN,
REFCLK => user_REFCLK,
REFCLK2 => user_REFCLK2,
REFCLKBSEL => user_REFCLKBSEL,
REFCLKSEL => user_REFCLKSEL,
RXBLOCKSYNC64B66BUSE => user_RXBLOCKSYNC64B66BUSE,
RXCOMMADETUSE => user_RXCOMMADETUSE,
RXDATANOSIZE => user_RXDATANOSIZE,
RXDEC64B66BUSE => user_RXDEC64B66BUSE,
RXDEC8B10BUSE => user_RXDEC8B10BUSE,
RXDESCRAM64B66BUSE => user_RXDESCRAM64B66BUSE,
RXIGNOREBTF => user_RXIGNOREBTF,
RXINTDATAWIDTH => user_RXINTDATAWIDTH,
RXN => user_RXN,
RXF => user_RXF,
RXPOLARITY => user_RXPOLARITY,
RXRESET => user_RXRESET,
RXSLIDE => user_RXSLIDE,
RXUSRCLK => user_RXUSRCLK,
RXUSRCLK2 => user_RXUSRCLK2,
TXBYPASS8B10B => user_TXBYPASS8B10B,
TXCHARDISPMODE => user_TXCHARDISPMODE,
TXCHARDISPVAL => user_TXCHARDISPVAL,
TXCHARISK => user_TXCHARISK,
TXDATA => user_TXDATA,
TXDATANOSIZE => user_TXDATANOSIZE,
TXENC64B66BUSE => user_TXENC64B66BUSE,
TXENC8B10BUSE => user_TXENC8B10BUSE,
TXXGEOBAK64B66BUSE => user_TXXGEOBAK64B66BUSE,
TXINHIBIT => user_TXINHIBIT,
TXINTDATAWIDTH => user_TXINTDATAWIDTH,
Verilog Instantiation Templates

GT10_PCI_EXPRESS_1

GT10_PCI_EXPRESS_1_name(

.TXPOLARITY => user_TXPOLARITY,
.TXRESET => user_TXRESET,
.TXSCRAM64B66BUSE => user_TXSCRAM64B66BUSE,
.TXUSRCLK => user_TXUSRCLK,
.TXUSRCLK2 => user_TXUSRCLK2);

.BREFCLKNOUT (user_BREFCLKNOUT),
.BREFCLKPOUT (user_BREFCLKPOUT),
.CHBONDDONE (user_CHBONDDONE),
.CHBONDO (user_CHBONDO),
.PMARKXLOCK (user_PMARKXLOCK),
.RXBUFSTATUS (user_RXBUFSTATUS),
.RXCHARISCOMMA (user_RXCHARISCOMMA),
.RXCHARISK (user_RXCHARISK),
.RXCLKCORCNT (user_RXCLKCORCNT),
.RXCOMMADET (user_RXCOMMADET),
.RXDATA (user_RXDATA),
.RXDISPERR (user_RXDISPERR),
.RXLOSSOFSYNC (user_RXLOSSOFSYNC),
.RXNOTINTABLE (user_RXNOTINTABLE),
.RXREALIGN (user_RXREALIGN),
.RXRECClk (user_RXRECClk),
.RXRUNDISP (user_RXRUNDISP),
.TXBUFERR (user_TXBUFERR),
.TXKERR (user_TXKERR),
.TXN (user_TXN),
.TXOUTCLK (user_TXOUTCLK),
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),
.BREFCLKNIN (user_BREFCLKNIN),
.BREFCLKPIN (user_BREFCLKPIN),
.CHBONDI (user_CHBONDI),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOBACK (user_LOOBACK),
.PMAINIT (user_PMAINIT),
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGRW (user_PMAREGRW),
.PMARESTROBE (user_PMARESTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARDISPVAL (user_TXCHARDISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 2;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001000101;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001000101;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 2;
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b0001111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "28_10";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

GT10_PCI_EXPRESS_2 GT10_PCI_EXPRESS_2_name(

.BREFCLKNOUT (user_BREFCLKNOUT),
.BREFCLKPOUT (user_BREFCLKPOUT),
.CHBONDDONE (user_CHBONDDONE),
.CHBONDO (user_CHBONDO),
.PMARKLOCK (user_PMARKLOCK),
.RXBUFSTATUS (user_RXBUFSTATUS),
.RXCHARISCOMMA (user_RXCHARISCOMMA),
.RXCHARISK (user_RXCHARISK),
.RXCLKCORCNT (user_RXCLKCORCNT),
.RXCOMMADET (user_RXCOMMADET),
.RXDATA (user_RXDATA),
.RXDISPER (user_RXDISPER),
.RXLOSOOSYNC (user_RXLOSOOSYNC),
.RXNOTINTABLE (user_RXNOTINTABLE),
.RXREALIGN (user_RXREALIGN),
.RXRECLK (user_RXRECLK),
.RXRUNDISP (user_RXRUNDISP),
.TXBUFERR (user_TXBUFERR),
.TXKERR (user_TXKERR),
.TXN (user_TXN),
.TXOUTCLK (user_TXOUTCLK),
.TXP (user_TXP),
.TXRUNDISP (user_TXRUNDISP),

.BREFCLKNIN (user_BREFCLKNIN),
.BREFCLKPIN (user_BREFCLKPIN),
.CHBONDI (user_CHBONDI),
.ENCHANSYNC (user_ENCHANSYNC),
.ENMCOMMAALIGN (user_ENMCOMMAALIGN),
.ENPCOMMAALIGN (user_ENPCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.PMAINIT (user_PMAINIT),
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),

1008 www.xilinx.com 1-800-255-7778
Libraries Guide ISE 6.1
.PMAREGRW (user_PMAREGRW),
.PMAREGSTROBE (user_PMAREGSTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (user_POWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMMADETUSE (user_RXCOMMADETUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXPASSB8B10B (user_TXPASSB8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARISMPVAL (user_TXCHARISMPVAL),
.TXCHARISK (user_TXCHARISK),
.TXDATA (user_TXDATA),
.TXDATAWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 2;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b000110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b0001111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "28_20";
defparam user_instance_name_RX_BUFFER_USE = "TRUE";
defparam user_instance_name_RX_LOS_INVALID_INCR = 1;
defparam user_instance_name_RX_LOS_THRESHOLD = 4;
defparam user_instance_name_RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name_TX_BUFFER_USE = "TRUE";

GT10_PCI_EXPRESS_4 GT10_PCI_EXPRESS_4_name(
    .BREFCLKNOUT (user_BREFCLKNOUT),
    .BREFCLKPOUT (user_BREFCLKPOUT),
    .CHBONDDONE (user_CHBONDDONE),
    .CHBONO (user_CHBONO),
    .PMARXLOCK (user_PMARXLOCK),
    .RXBUFSTATUS (user_RXBUFSTATUS),
    .RXCHARISCOMMA (user_RXCHARISCOMMA),
    .RXCHARISK (user_RXCHARISK),
    .RXCLKCORCNT (user_RXCLKCORCNT),
    .RXCOMMADET (user_RXCOMMADET),
    .RXDATA (user_RXDATA),
    .RXDISPERR (user_RXDISPERR),
    .RXLOSSOFSYNC (user_RXLOSSOFSYNC),
    .RXNOTINTABLE (user_RXNOTINTABLE),
    .RXREALIGN (user_RXREALIGN),
    .RXRECLK (user_RXRECLK),
    .RXRUNDISP (user_RXRUNDISP),
    .TXBUFERR (user_TXBUFERR),
    .TXXERR (user_TXXERR),
    .TXN (user_TXN),
    .TXOUTCLK (user_TXOUTCLK),
    .TXP (user_TXP),
)}
.TXRUNDISP (user_TXRUNDISP),
.BREFCLKIN (user_BREFCLKIN),
.BREFCLKPIN (user_BREFCLKPIN),
.CHANBOND (user_CHANBOND),
.CHANENCSYN (user_CHANENCSYN),
.ENMDCOMMAALIGN (user_ENMDCOMMAALIGN),
.ENPMCOMMAALIGN (user_ENPMCOMMAALIGN),
.LOOPBACK (user_LOOPBACK),
.PMAIN (user_PMAIN),
.PMAREGADDR (user_PMAREGADDR),
.PMAREGDATAIN (user_PMAREGDATAIN),
.PMAREGWR (user_PMAREGWR),
.PMAREGSTROBE (user_PMAREGSTROBE),
.PMARXLOCKSEL (user_PMARXLOCKSEL),
.POWERDOWN (userPOWERDOWN),
.REFCLK (user_REFCLK),
.REFCLK2 (user_REFCLK2),
.REFCLKBSEL (user_REFCLKBSEL),
.REFCLKSEL (user_REFCLKSEL),
.RXBLOCKSYNC64B66BUSE (user_RXBLOCKSYNC64B66BUSE),
.RXCOMPADTEUSE (user_RXCOMPADTEUSE),
.RXDATAWIDTH (user_RXDATAWIDTH),
.RXDEC64B66BUSE (user_RXDEC64B66BUSE),
.RXDEC8B10BUSE (user_RXDEC8B10BUSE),
.RXDESCRAM64B66BUSE (user_RXDESCRAM64B66BUSE),
.RXIGNOREBTF (user_RXIGNOREBTF),
.RXINTDATAWIDTH (user_RXINTDATAWIDTH),
.RXN (user_RXN),
.RXP (user_RXP),
.RXPOLARITY (user_RXPOLARITY),
.RXRESET (user_RXRESET),
.RXSLIDE (user_RXSLIDE),
.RXUSRCLK (user_RXUSRCLK),
.RXUSRCLK2 (user_RXUSRCLK2),
.TXBYPASS8B10B (user_TXBYPASS8B10B),
.TXCHARDISPMODE (user_TXCHARDISPMODE),
.TXCHARISPVAL (user_TXCHARISPVAL),
.TXCHARISK (user_TXCHARISK),
.TXD (user_TXDATA),
.TXDARWIDTH (user_TXDATAWIDTH),
.TXENC64B66BUSE (user_TXENC64B66BUSE),
.TXENC8B10BUSE (user_TXENC8B10BUSE),
.TXGEARBOX64B66BUSE (user_TXGEARBOX64B66BUSE),
.TXINHIBIT (user_TXINHIBIT),
.TXINTDATAWIDTH (user_TXINTDATAWIDTH),
.TXPOLARITY (user_TXPOLARITY),
.TXRESET (user_TXRESET),
.TXSCRAM64B66BUSE (user_TXSCRAM64B66BUSE),
.TXUSRCLK (user_TXUSRCLK),
.TXUSRCLK2 (user_TXUSRCLK2));

defparam user_instance_name.ALIGN_COMMA_WORD = 2;
defparam user_instance_name.CHAN_BOND_LIMIT = 16;
defparam user_instance_name.CHAN_BOND_MODE = "OFF";
defparam user_instance_name.CHAN_BOND_ONE_SHOT = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_1_1 = 11'b0011011100;
defparam user_instance_name.CHAN_BOND_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_1_3 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_4 = 11'b00001001010;
defparam user_instance_name.CHAN_BOND_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_1 = 11'b00110111100;
defparam user_instance_name.CHAN_BOND_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CHAN_BOND_SEQ_2_3 = 11'b00001000101;
defparam user_instance_name.CHAN_BOND_SEQ_2_4 = 11'b00001000101;
defparam user_instance_name.CHAN_BOND_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CHAN_BOND_SEQ_2_USE = "FALSE";
defparam user_instance_name.CHAN_BOND_SEQ_LEN = 2;
defparam user_instance_name.CLK_COR_8B10B_DE = "FALSE";
defparam user_instance_name.CLK_COR_MAX_LAT = 36;
defparam user_instance_name.CLK_COR_MIN_LAT = 28;
defparam user_instance_name.CLK_COR_SEQ_1_1 = 11'b00100011100;
defparam user_instance_name.CLK_COR_SEQ_1_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_1_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_1 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_2 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_3 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_4 = 11'b00000000000;
defparam user_instance_name.CLK_COR_SEQ_2_MASK = 4'b0000;
defparam user_instance_name.CLK_COR_SEQ_2_USE = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_DROP = "FALSE";
defparam user_instance_name.CLK_COR_SEQ_LEN = 2;
defparam user_instance_name.CLK_CORRECT_USE = "TRUE";
defparam user_instance_name.COMMA_10B_MASK = 10'b00011111111;
defparam user_instance_name.DEC_MCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_PCOMMA_DETECT = "TRUE";
defparam user_instance_name.DEC_VALID_COMMA_ONLY = "TRUE";
defparam user_instance_name.MCOMMA_10B_VALUE = 10'b1010000011;
defparam user_instance_name.MCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_10B_VALUE = 10'b0101111100;
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PCOMMA_DETECT = "TRUE";
defparam user_instance_name.PMA_PWR_CNTRL = 8'b11111111;
defparam user_instance_name.PMA_SPEED = "28_40";
defparam user_instance_name.RX_BUFFER_USE = "TRUE";
defparam user_instance_name.RX_LOS_INVALID_INCR = 1;
defparam user_instance_name.RX_LOS_THRESHOLD = 4;
defparam user_instance_name.RX_LOSS_OF_SYNC_FSM = "TRUE";
defparam user_instance_name.TX_BUFFER_USE = "TRUE";

Commonly Used Constraints

None
### Single- and Multiple-Input Buffers

#### IBUF

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
<td>IBUF, 4, 8, 16</td>
</tr>
</tbody>
</table>

#### IBUF4

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
<td>IBUF4</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td>IBUF4</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
<td>IBUF4</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
<td>IBUF4</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
<td>IBUF4</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
<td>IBUF4</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td>IBUF4</td>
</tr>
</tbody>
</table>

#### IBUF8, IBUF16

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
<td>IBUF8, IBUF16</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td>IBUF8, IBUF16</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
<td>IBUF8, IBUF16</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
<td>IBUF8, IBUF16</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
<td>IBUF8, IBUF16</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
<td>IBUF8, IBUF16</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
<td>IBUF8, IBUF16</td>
</tr>
</tbody>
</table>

IBUF, IBUF4, IBUF8, and IBUF16 are single- and multiple-input buffers. An IBUF isolates the internal circuit from the signals coming into a chip. IBUFs are contained in input/output blocks (IOBs). IBUF inputs (I) are connected to an IPAD or an IOPAD. IBUF outputs (O) are connected to the internal circuit.

For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, see “IBUF_selectIO” for information on IBUF variants with selectable I/O interfaces.

Usage

IBUFs are typically inferred for all top level input ports, but they can also be instantiated if necessary.

VHDL Instantiation Template

```vhdl
-- Component Declaration for IBUF should be placed
-- after architecture statement but before begin keyword

component IBUF
    port (O : out STD_ULOGIC;
         I : in STD_ULOGIC);
end component;

-- Component Attribute specification for IBUF
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IBUF should be placed
```
-- in architecture after the begin keyword

IBUF_INSTANCE_NAME : IBUF
    port map (O => user_O,
                I => user_I);

Verilog Instantiation Template

IBUF instance_name (.O (user_O),
                    .I (user_I));

Commonly Used Constraints

BUFG (CPLDs)
IOSTANDARD
IOBDELAY
**IBUF_selectIO**

Single Input Buffer with Selectable I/O Interface

Architectures Supported

<table>
<thead>
<tr>
<th>IBUF_selectIO</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td></td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td></td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td></td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

For Spartan-II, Spartan-IIE, Virtex, and Virtex-E, IBUF and its selectIO variants (listed in the "Components" column in the following table) are single input buffers whose I/O interface corresponds to a specific I/O standard. The name extensions (LVCMOS2, PCI33_3, PCI33_5, etc.) specify the standard. For example, IBUF_SSTL3_II is a single input buffer that uses the SSTL3_II I/O-signaling standard. You can attach an IOSTANDARD attribute to an IBUF instance instead of using an IBUF_selectIO component. Check marks (√) in the "Spartan-II, Virtex" and "Spartan-IIE, Virtex-E" columns indicate the components and IOSTANDARD attribute values available for those architectures.

An IBUF isolates the internal circuit from the signals coming into a chip. For Spartan-II, Spartan-IIE, Virtex, and Virtex-E, the dedicated GCLKIOB pad is input only. IBUF inputs (I) are connected to an IPAD or IOPAD. IBUF outputs (O) are connected to the internal circuit.

The hardware implementation of the I/O standards requires that you follow a set of usage rules for the SelectIO buffers. See “SelectIO Usage Rules” below for information on using these components and IOSTANDARD attributes.

### Spartan-II, Spartan-IIE, Virtex, and Virtex-E IBUF_selectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-II, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>VREF</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUF</td>
<td>√</td>
<td>√</td>
<td>(defaults to LVTTL)</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUF_AGPP</td>
<td>√</td>
<td>√</td>
<td>AGP</td>
<td>1.32</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_CTT</td>
<td>√</td>
<td>√</td>
<td>CTT</td>
<td>1.50</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_GTLP</td>
<td>√</td>
<td>√</td>
<td>GTLP</td>
<td>0.80</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_HSTL_I</td>
<td>√</td>
<td>√</td>
<td>HSTL_I</td>
<td>0.75</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_HSTL_II</td>
<td>√</td>
<td>√</td>
<td>HSTL_II</td>
<td>0.90</td>
<td>1.5</td>
</tr>
<tr>
<td>IBUF_HSTL_III</td>
<td></td>
<td>√</td>
<td>HSTL_III</td>
<td>0.90</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_LVCMOS2</td>
<td>√</td>
<td>√</td>
<td>LVCMOS2</td>
<td>No</td>
<td>2.5</td>
</tr>
</tbody>
</table>
The Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X library includes some IBUF_selectIO components for compatibility with older, existing designs and other architectures. For new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs, however, the recommended method for using IBUF SelectIO buffers is to attach an IOSTANDARD attribute to an IBUF component. For example, attach IOSTANDARD=GTLP to an IBUF instead of using the IBUF_GTLP component for new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs. The IOSTANDARD attributes that can be attached to an IBUF component are listed in the "IOSTANDARD (Attribute Value)" column in the following table. See “SelectIO Usage Rules” for information on using these IOSTANDARD attributes.

Attach an IOSTANDARD attribute to an IBUF and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the input for the I/O standard associated with that value.

### Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X IBUF_selectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-III, Virtex</th>
<th>Spartan-II, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>VREF</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUF_LVCMOS18</td>
<td>√</td>
<td></td>
<td>LVCMOS18</td>
<td>No</td>
<td>1.8</td>
</tr>
<tr>
<td>IBUF_LVDS</td>
<td>√</td>
<td></td>
<td>LVDS</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_LVPECL</td>
<td>√</td>
<td></td>
<td>LVPECL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_PCI33_3</td>
<td>√</td>
<td>√</td>
<td>PCI33_3</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUF_PCI33_5</td>
<td>√</td>
<td></td>
<td>PCI33_5</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_PCI66_3</td>
<td>√</td>
<td></td>
<td>PCI66_3</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUF_PCIE66_3</td>
<td>√</td>
<td></td>
<td>PCIE66_3</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUF_SSTL2_I</td>
<td>√</td>
<td>√</td>
<td>SSTL2_I</td>
<td>1.25</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_SSTL2_II</td>
<td>√</td>
<td></td>
<td>SSTL2_II</td>
<td>1.25</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_SSTL3_I</td>
<td>√</td>
<td></td>
<td>SSTL3_I</td>
<td>1.50</td>
<td>No</td>
</tr>
<tr>
<td>IBUF_SSTL3_II</td>
<td>√</td>
<td></td>
<td>SSTL3_II</td>
<td>1.50</td>
<td>No</td>
</tr>
</tbody>
</table>

### Spartan-II, Virtex, Virtex-E IBUF selectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-III, Virtex</th>
<th>Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>VREF</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTL</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTL_DCI</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTLP</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTLP_DCI</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTL_I</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTL_II</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X IBUF_selectIO IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Architectures</th>
<th>Attribute Values</th>
<th>Termination Type</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTL_II_18</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>None</td>
<td>0.9</td>
<td>No</td>
</tr>
<tr>
<td>HSTL_II_DCI</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>Split</td>
<td>0.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_II_DCI_18</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>Split</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_III</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>None</td>
<td>0.90</td>
<td>No</td>
</tr>
<tr>
<td>HSTL_III_18</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>None</td>
<td>1.10</td>
<td>No</td>
</tr>
<tr>
<td>HSTL_III_DCI</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>Single</td>
<td>0.90</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_III_DCI_18</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>Single</td>
<td>1.10</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_IV</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>None</td>
<td>0.90</td>
<td>No</td>
</tr>
<tr>
<td>HSTL_IV_18</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>None</td>
<td>1.10</td>
<td>No</td>
</tr>
<tr>
<td>HSTL_IV_DCI</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>Single</td>
<td>0.90</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_IV_DCI_18</td>
<td>Spartan-3, Virtex-II, Virtex-II Pro X</td>
<td>Single</td>
<td>1.10</td>
<td>1.8</td>
</tr>
<tr>
<td>LVCMOS12</td>
<td>None</td>
<td>None</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVCMOS15</td>
<td>None</td>
<td>None</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>LVCMOS18</td>
<td>None</td>
<td>None</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>LVCMOS25</td>
<td>None</td>
<td>None</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>LVCMOS33</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>LVDCI_15</td>
<td>None</td>
<td>None</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>LVDCI_18</td>
<td>None</td>
<td>None</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>LVDCI_25</td>
<td>None</td>
<td>None</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>LVDCI_33</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_15</td>
<td>None</td>
<td>None</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_18</td>
<td>None</td>
<td>None</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_25</td>
<td>None</td>
<td>None</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_33</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>VTTTL(default)</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>PCI33_3</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>PCI66_3</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>PCIIX</td>
<td>None</td>
<td>None</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>SSTL18_1</td>
<td>None</td>
<td>None</td>
<td>0.9</td>
<td>No</td>
</tr>
<tr>
<td>SSTL18_1_DCI</td>
<td>None</td>
<td>Split</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>None</td>
<td>None</td>
<td>0.9</td>
<td>No</td>
</tr>
<tr>
<td>SSTL18_II_DCI</td>
<td>None</td>
<td>Split</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL2_1</td>
<td>None</td>
<td>None</td>
<td>1.25</td>
<td>No</td>
</tr>
<tr>
<td>SSTL2_1_DCI</td>
<td>None</td>
<td>Split</td>
<td>1.25</td>
<td>2.5</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>None</td>
<td>None</td>
<td>1.25</td>
<td>No</td>
</tr>
<tr>
<td>SSTL2_II_DCI</td>
<td>None</td>
<td>Split</td>
<td>1.25</td>
<td>2.5</td>
</tr>
<tr>
<td>SSTL3_1</td>
<td>None</td>
<td>None</td>
<td>1.50</td>
<td>No</td>
</tr>
<tr>
<td>SSTL3_1_DCI</td>
<td>None</td>
<td>Split</td>
<td>1.50</td>
<td>3.3</td>
</tr>
</tbody>
</table>
SelectIO Usage Rules

The Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X architectures include a versatile SelectIO interface to multiple voltage and drive standards. To select an I/O standard, you must choose the appropriate component from the library or add an IOSTANDARD attribute to the appropriate buffer component. For example, for an input buffer that uses the GTL standard, you would choose the IBUF_GTL component or choose the IBUF component and attach the IOSTANDARD=GTL attribute to it.

See the following sections for information on the various input/output buffer components and attributes available to implement the desired standard:

- “IBUF_selectIO”
- “IBUFG, IBUFG_selectIO”
- “IOBUF, IOBUF_selectIO”
- “OBUF_selectIO”
- “OBUFT_selectIO”

The hardware implementation of the various I/O standards requires that certain usage rules be followed. Each I/O standard has voltage source requirements for input reference (VREF), output drive (VCCO), or both. In addition, Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X have terminate type requirements. Each Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X device has eight banks (two on each edge). Each bank has voltage sources shared by all I/O in the bank. Therefore, in a particular bank, the voltage source (for either input or output) must be of the same type.

- For Spartan-II, Spartan-IIE, Virtex, and Virtex-E, see “Virtex, Virtex-E, Spartan-II, and Spartan-IIE Banking Rules” below. Virtex-E follows the same banking rules as Virtex with a few additions.
- See “Additional Banking Rules for Virtex-E and Spartan-II” below for the additional Virtex-E rules. Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X have their own set of banking rules.

Virtex, Virtex-E, Spartan-II, and Spartan-IIE Banking Rules

The hardware implementation of the various I/O standards requires that certain usage rules be followed. As shown in the following table, each I/O standard has voltage source requirements for input reference (VREF), output drive (VCCO), or

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Input</th>
<th>VREF Input</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTL3_II</td>
<td></td>
<td></td>
<td></td>
<td>None</td>
<td>1.50</td>
<td>No</td>
</tr>
<tr>
<td>SSTL3_II_DCI</td>
<td></td>
<td></td>
<td></td>
<td>Split</td>
<td>1.50</td>
<td>3.3</td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.
both. Each Spartan-II, Spartan-IIE, Virtex, and Virtex-E device has eight banks (two on each edge). Each bank has voltage sources shared by all I/O in the bank. Therefore, in a particular bank, the voltage source (for either input or output) must be of the same type. The Input Banking (VREF) Rules section and the Output Banking (VCCO) Rules section below summarize the SelectIO component usage rules based on the hardware implementation.

### I/O Standards Supported in Virtex, Virtex-E, Spartan-II, and Spartan-IIE

<table>
<thead>
<tr>
<th>I/O Standard</th>
<th>Application</th>
<th>Description</th>
<th>Output VCCO</th>
<th>Input VCCO</th>
<th>VREF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGP</td>
<td>Graphics</td>
<td>Advanced graphics port</td>
<td>3.3</td>
<td>No</td>
<td>1.32</td>
</tr>
<tr>
<td>CTT</td>
<td>Memory</td>
<td>Center tap terminated</td>
<td>3.3</td>
<td>No</td>
<td>1.50</td>
</tr>
<tr>
<td>LVTTL</td>
<td>General Purpose</td>
<td>Low voltage transistor-transistor logic</td>
<td>3.3</td>
<td>3.3*</td>
<td>No</td>
</tr>
<tr>
<td>LVCMOS2</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
<td>2.5</td>
<td>2.5*</td>
<td>No</td>
</tr>
<tr>
<td>PCI33_3</td>
<td>PCI</td>
<td>Peripheral component interface (33MHz 3.3V)</td>
<td>3.3</td>
<td>3.3*</td>
<td>No</td>
</tr>
<tr>
<td>PCI33_5</td>
<td>PCI</td>
<td>Peripheral component interface (33MHz 5.0V)</td>
<td>3.3</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PCI66_3</td>
<td>PCI</td>
<td>Peripheral component interface (66MHz 3.3V)</td>
<td>3.3</td>
<td>3.3*</td>
<td>No</td>
</tr>
<tr>
<td>GTL</td>
<td>Backplane</td>
<td>Gunning transceiver logic interface (to processors or backplane driver)</td>
<td>No</td>
<td>No</td>
<td>0.80</td>
</tr>
<tr>
<td>GTL+ (GTLP)</td>
<td>Backplane</td>
<td>Gunning transceiver logic interface Plus</td>
<td>No</td>
<td>No</td>
<td>1.00</td>
</tr>
<tr>
<td>HSTL_I</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
<td>1.5</td>
<td>No</td>
<td>0.75</td>
</tr>
<tr>
<td>HSTL_III</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
<td>1.5</td>
<td>No</td>
<td>0.90</td>
</tr>
<tr>
<td>HSTL_IV</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
<td>1.5</td>
<td>No</td>
<td>0.90</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
<td>2.5</td>
<td>No</td>
<td>1.25</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
<td>2.5</td>
<td>No</td>
<td>1.25</td>
</tr>
<tr>
<td>SSTL3_I</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
<td>3.3</td>
<td>No</td>
<td>1.50</td>
</tr>
<tr>
<td>SSTL3_II</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
<td>3.3</td>
<td>No</td>
<td>1.50</td>
</tr>
</tbody>
</table>

*Only LVTTL, LVCMOS, and PCI need Input VCCO in Virtex-E and Spartan-IIE parts.

### Input Banking (VREF) Rules

The low-voltage I/O standards that have a differential amplifier input require a voltage reference input (VREF). The VREF voltage source is provided as an external signal to the chip.

- Any input buffer component that does not require a VREF source (LVTTL, LVCMOS2, PCI) can be placed in any bank.
• All input buffer components that require a VREF source (GTL*, HSTL*, SSTL*, CTT, AGP) must be of the same I/O standard in a particular bank. For example, IBUF_SSTL2_I and IBUF_SSTL2_I are compatible since they are the same I/O standard (SSTL2_I).

• If the bank contains any input buffer component that requires a VREF source, the following conditions apply.
  ♦ One or more VREF sources must be connected to the bank via an IOB.
  ♦ The number of VREF sources is dependent on the device and package.
  ♦ The locations of the VREF sources are fixed for each device/package.
  ♦ All VREF sources must be used in that bank.

• If the bank contains no input buffer component that requires a VREF source, the IOBs for VREF sources can be used for general I/O.

• Output buffer components of any type can be placed in the bank.

Output Banking (VCCO) Rules

Because Virtex, Virtex-E, Spartan-II, and Spartan-IIE have multiple low-voltage standards, some control is required over the distribution of VCCO, the drive source voltage for output pins. To provide for maximum flexibility, the output pins are banked. In comparison to the VREF sources described above, the VCCO voltage sources are dedicated pins on the device and do not consume valuable IOBs.

• Any output buffer component that does not require a VCCO source (GTL, GTL+) can be placed in any bank.

• To be placed in a particular bank, all output buffer components that require VCCO must have the same supply voltage (VCCO). For example, OBUF_SSTL3_I and OBUF_PCI33_3 are compatible in the same output bank since VCCO=3.3 for both.

• Input buffer components of any type can be placed in the bank.

• The configuration pins on a Virtex, Virtex-E, Spartan-II, and Spartan-IIE device are on the right side of the chip. When configuring the device through a serial ProM, the user is required to use a VCCO of 3.3V in the two banks on the right hand side of the chip. If the user is not configuring the device through a serial ProM, the VCCO requirement is dependent upon the configuration source.

Banking Rules for OBUFT_selectIO with KEEPER

If a KEEPER symbol is attached to an OBUFT_selectIO component (3-state output buffer) for an I/O standard that requires a VREF (for example, OBUFT_GTL, OBUFT_SSTL3_I), then the OBUFT_selectIO component follows the same rules as an IOBUF_selectIO component for the same standard. It must follow both the input banking and output banking rules. The KEEPER element requires that the VREF be properly driven.

Additional Banking Rules for Virtex-E and Spartan-IIE

The Virtex-E and Spartan-IIE architectures require the same banking rules as described in the “Virtex, Virtex-E, Spartan-II, and Spartan-IIE Banking Rules” section.
Additional I/O standards are supported as indicated in the following table.

### Additional I/O Standards Supported in Virtex-E and Spartan-IIE

<table>
<thead>
<tr>
<th>I/O Standard</th>
<th>Application</th>
<th>Description</th>
<th>Output VCCO</th>
<th>Input VCCO</th>
<th>VREF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVCMOS18</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
<td>1.8</td>
<td>1.8</td>
<td>No</td>
</tr>
<tr>
<td>LVPECL</td>
<td>High performance clocking, backplanes, differential 100MHz+ clocking, optical transceiver, high speed networking and mixed-signal interfacing</td>
<td>Low voltage positive emitter couple logic</td>
<td>2.5</td>
<td>2.5</td>
<td>No</td>
</tr>
</tbody>
</table>

### Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X Banking Rules

The hardware implementation of the various I/O standards requires that certain usage rules be followed. Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X devices have eight banks (two on each edge), numbered from 0 through 7. Each bank has voltage sources shared by all I/O in the bank.

The following table describes each I/O standard.

### Descriptions of I/O Standards Supported In Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X

<table>
<thead>
<tr>
<th>I/O Standard</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGPc</td>
<td>Graphics</td>
<td>Advanced graphics port</td>
</tr>
<tr>
<td>GTL</td>
<td>Memory</td>
<td>Gunning transceiver logic interface (to processors or backplace driver)</td>
</tr>
<tr>
<td>GTL_DCI</td>
<td>Memory</td>
<td>Gunning transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>GTL+ (GTLP)</td>
<td>Memory</td>
<td>Gunning transceiver logic interface Plus</td>
</tr>
<tr>
<td>GTLP_DCI</td>
<td>Memory</td>
<td>Gunning transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_I</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_II</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_II_18</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_II_DCI</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_II_DCI_18</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_III</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
</tbody>
</table>
### Descriptions of I/O Standards Supported In Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X

<table>
<thead>
<tr>
<th>I/O Standard</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTL_III_18</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_III__DCI</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_III__DCI_18</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_IVb</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_IV_18b</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic</td>
</tr>
<tr>
<td>HSTL_IV__DCIb</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>HSTL_IV__DCI_18b</td>
<td>Hitachi SRAM</td>
<td>High Speed transceiver logic interface with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVCMOS12ad</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
</tr>
<tr>
<td>LVCMOS15a</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
</tr>
<tr>
<td>LVCMOS18a</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
</tr>
<tr>
<td>LVCMOS25a</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
</tr>
<tr>
<td>LVCMOS33ab</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor</td>
</tr>
<tr>
<td>LVDCI_15</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_18</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_25</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_33b</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_DV2_15</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_DV2_18</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_DV2_25</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVDCI_DV2_33b</td>
<td>General Purpose</td>
<td>Low voltage complementary metal-oxide semiconductor with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>LVTTLab</td>
<td>General Purpose</td>
<td>Low voltage transistor-transistor logic</td>
</tr>
<tr>
<td>PCI33_3b</td>
<td>PCI</td>
<td>Peripheral component interface (33MHz 3.3V)</td>
</tr>
<tr>
<td>PCI66_3b</td>
<td>PCI</td>
<td>Peripheral component interface (66MHz 3.3V)</td>
</tr>
<tr>
<td>PCIXb</td>
<td>PCI</td>
<td>Peripheral component interface</td>
</tr>
<tr>
<td>SSTL18_I</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
</tr>
<tr>
<td>SSTL18_I__DCI</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
</tr>
<tr>
<td>SSTL18_II__DCIb</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM with on-chip Digital Controlled Impedance</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>Synchronous DRAM</td>
<td>Stub-series terminated logic interface for SDRAM</td>
</tr>
</tbody>
</table>
The following rules apply for using the various IO standards with Spartan-3, Virtex-II, Virtex-II Pro, or Virtex-II Pro X:

- In any particular Spartan-3, Virtex-II, Virtex-II Pro, or Virtex-II Pro X I/O bank, the voltage sources (both input and output) must be compatible. That is, they must have either the same voltage or an undefined (No) voltage.
- VREF, VCCO input, and VCCO output must be compatible within an I/O bank.
- In addition, to VREF and VCCO compatibility, the terminate type I/O standards must be compatible within the bank.

For terminate type compatibility, the following rules apply:

- Only one I/O buffer with terminate type of SINGLE can be in a particular bank.
- Only one I/O buffer with terminate type of SPLIT can be in a particular bank.
- Multiple I/O buffers with NONE and DRIVER terminate types can be in a particular bank.
- SPLIT and SINGLE can co-exist in the same bank.
- NONE and DRIVER types can co-exist with SPLIT and SINGLE types.

- The bottom edge of a Spartan-3, Virtex-II Pro, or Virtex-II Pro X device is set for 3.3V. Therefore, on the bottom edge of the device, VCCO output and VCCO input must be 3.3V or No.
- To place an I/O buffer that requires a VREF in a bank, the reserved VREF sites in that bank must be empty.
- To place an I/O buffer that has a terminate type of SINGLE, SPLIT, or DRIVER in a bank, the reserved VREF sites in that bank must be empty.
- For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, differential signaling standards apply to IBUFDS, IBUFGDS, IBUFGDS_DIFF_OUT, OBUFDS, and OBUFTDS only (not IBUF or OBUF).
The following table summarizes the values that you need to check for compatibility for each combination of I/O buffer programming (input, output, or bidirectional buffer). For example, the table shows that if you configure an output buffer as LVCMOS25, which has an output voltage of 2.5V, and an input buffer as LVCMOS15, which as an input voltage of 1.5V, the Out/In Voltage is checked. Because they have different voltages, this combination would not be allowed in a particular I/O bank.

<table>
<thead>
<tr>
<th>IOB Programming Combinations</th>
<th>VREF</th>
<th>Output VCCO</th>
<th>Input VCCO</th>
<th>Out/In Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Input</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Output</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Bidirectional</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Input</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Output</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Bidirectional</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Bidirectional</td>
<td>Input</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Bidirectional</td>
<td>Output</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
<tr>
<td>Bidirectional</td>
<td>Bidirectional</td>
<td>Check</td>
<td>Check</td>
<td>Check</td>
</tr>
</tbody>
</table>

**Usage**

The recommended usage for IBUF_selectIO is to allow the IBUFs be inferred and apply the IOSTANDARD constraint to the input in either the UCF or in the HDL code. IBUF_selectIO can also be instantiated if necessary.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for IBUF_selectIO should be placed
-- after architecture statement but before begin keyword

component IBUF_selectIO
    port (O : out STD_ULOGIC;
          I : in STD_ULOGIC);
end component;

-- Component Attribute specification for IBUF_selectIO
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IBUF_selectIO should be placed
-- in architecture after the begin keyword

IBUF_selectIO_INSTANCE_NAME : IBUF_selectIO
    port map (O => user_O,
              I => user_I);
```

**Verilog Instantiation Template**

```verilog
IBUF_selectIO instance_name (.O (user_O),
                              .I (user_I));
```
Commonly Used Constraints

IOBDELAY, IOSTANDARD
IBUFDS

Differential Signaling Input Buffer with Selectable I/O Interface

Architectures Supported

<table>
<thead>
<tr>
<th>IBUFDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

IBUFDS is an input buffer that supports low-voltage, differential signaling. In IBUFDS, a design level interface signal is represented as two distinct ports (I and IB), one deemed the “master” and the other the “slave.” The master and the slave are opposite phases of the same logical signal (for example, MYNET and MYNETB).

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>IB</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* The dash (-) means No Change.

The IOSTANDARD attribute values listed in the following table can be applied to an IBUFDS component to provide selectIO interface capability.

A separate SelectIO component is not provided. Attach an IOSTANDARD attribute to an IBUFDS and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the inputs for the I/O standard associated with that value.
## Usage

For HDL, this design element is supported for instantiation but not for inference.

### VHDL Instantiation Template

```vhdl
-- Component Declaration for IBUFDS should be placed
-- after architecture statement but before begin keyword

component IBUFDS
    port (O : out STD_ULOGIC;
          I : in STD_ULOGIC;
          IB : in STD_ULOGIC);
end component;

-- Component Attribute specification for IBUFDS
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IBUFDS should be placed
-- in architecture after the begin keyword

IBUFDS_INSTANCE_NAME : IBUFDS
    port map (O => user_O,
              I => user_I,
              IB => user_IB);
```

### Verilog Instantiation Template

```verilog
IBUFDS instance_name (.O (user_O),
                       .I (user_I),
                       .IB (user_IB));
```
Commonly Used Constraints

IOSTANDARD

IOBDELAY
IBUFG, IBUFG\_selectIO

Dedicated Input Buffer with Selectable I/O Interface

Architectures Supported

<table>
<thead>
<tr>
<th>IBUFG, IBUFG_selectIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

For Virtex, Virtex-E, Spartan-II, and Spartan-IIE, IBUFG and its selectIO variants (listed in the "Components" column in the table below) are dedicated input buffers for connecting to the clock buffer BUFG or CLKDLL. The name extensions (LVCMOS2, PCI33\_3, PCI33\_5, etc.) specify the standard. For example, IBUFG\_SSTL3\_II is a single input buffer that uses the SSTL3\_II I/O-signaling standard. You can attach an IOSTANDARD attribute to an IBUFG instance instead of using an IBUFG\_selectIO component. Check marks (\(\checkmark\)) in the "Spartan-II, Virtex" and "Spartan-IIE, Virtex-E" columns indicate the components and IOSTANDARD attribute values available for those architectures.

The Xilinx implementation software converts each BUFG to an appropriate type of global buffer for the target PLD device. The IBUFG input can only be driven by the global clock pins. The IBUFG output can drive CLKin of a DLL/DCM, BUFG, or user logic. IBUFG can be routed to user logic and does not have to be routed to a DLL. The IBUFG can only be driven by an IPAD.

The hardware implementation of the I/O standards requires that you follow a set of usage rules for the SelectIO buffers. See the "SelectIO Usage Rules" section included in the IBUFG\_selectIO section for information on using these components and the IOSTANDARD attributes.

Spartan-II, Spartan-IIE, Virtex, and Virtex-E IBUFG\_selectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-IIE, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>VREF</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUFG</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(defaults to LVTTL)</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUFG_AGP</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>AGP</td>
<td>1.32</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_CTT</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>CTT</td>
<td>1.50</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_GTL</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>GTL</td>
<td>0.80</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_GTLP</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>GTLP</td>
<td>1.00</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_HSTL_I</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>HSTL_I</td>
<td>0.75</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_HSTL_III</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>HSTL_III</td>
<td>0.90</td>
<td>1.5</td>
</tr>
<tr>
<td>IBUFG_HSTL_IV</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>HSTL_IV</td>
<td>0.90</td>
<td>No</td>
</tr>
</tbody>
</table>
Spartan-II, Spartan-IIIE, Virtex, and Virtex-E IBUFSelectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-IIIE, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>VREF</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUFGLVCMOS2</td>
<td>√</td>
<td>√</td>
<td>LVCMOS2</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>IBUFGLVCMOS18</td>
<td>√</td>
<td>√</td>
<td>LVCMOS18</td>
<td>No</td>
<td>1.8</td>
</tr>
<tr>
<td>IBUFGLVDS</td>
<td>√</td>
<td>√</td>
<td>LVDS</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFGLVPECL</td>
<td>√</td>
<td>√</td>
<td>LVPECL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_PCI33_3</td>
<td>√</td>
<td>√</td>
<td>PCI33_3</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUFG_PCI33_5</td>
<td>√</td>
<td>√</td>
<td>PCI33_5</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_PCI66_3</td>
<td>√</td>
<td>√</td>
<td>PCI66_3</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUFG_PCIEX66_3</td>
<td>√</td>
<td>√</td>
<td>PCIeX66_3</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>IBUFG_SSTL2_I</td>
<td>√</td>
<td>√</td>
<td>SSTL2_I</td>
<td>1.25</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_SSTL2_II</td>
<td>√</td>
<td>√</td>
<td>SSTL2_II</td>
<td>1.25</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_SSTL3_I</td>
<td>√</td>
<td>√</td>
<td>SSTL3_I</td>
<td>1.50</td>
<td>No</td>
</tr>
<tr>
<td>IBUFG_SSTL3_II</td>
<td>√</td>
<td>√</td>
<td>SSTL3_II</td>
<td>1.50</td>
<td>No</td>
</tr>
</tbody>
</table>

*Not supported for Virtex-E.

The Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X library includes some IBUFSelectIO components for compatibility with older, existing designs and other architectures. For new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs, however, the recommended method for using IBUF SelectIO buffers is to attach an IOSTANDARD attribute to an IBUF component. For example, attach IOSTANDARD=GTLP to an IBUF instead of using the IBUFG_GTLP component for new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs. The IOSTANDARD attributes that can be attached to an IBUF component are listed in the "IOSTANDARD (Attribute Value)" column in the following table. See the "SelectIO Usage Rules" section for information on using these IOSTANDARD attributes.

Attach an IOSTANDARD attribute to an IBUF and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the input for the I/O standard associated with that value.

Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X IBUFSelectIO IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Input</th>
<th>VREF Input *</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGP</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>None</td>
<td>1.32</td>
<td>No</td>
</tr>
<tr>
<td>GTL</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>0.80</td>
<td>No</td>
</tr>
<tr>
<td>GTL_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Single</td>
<td>0.80</td>
<td>1.2</td>
</tr>
<tr>
<td>GTLP</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>1.00</td>
<td>No</td>
</tr>
<tr>
<td>GTLP_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Single</td>
<td>1.00</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>0.75</td>
<td>No</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>0.9</td>
<td>No</td>
</tr>
</tbody>
</table>
## Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X IBUFG_selectIO IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Architectures</th>
<th>Attribute Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Virtex-II</td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_II_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_II_DCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_III</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_III_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_III_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_III_DCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV_DCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVCMOS18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVCMOS25</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVCMOS33</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_15</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_25</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_33</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_15</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_25</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_33</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVTTL (default)</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PCI33_3</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PCI66_3</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PCIX</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_I_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_II_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_I_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_II_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL3_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL3_I_DCI</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X IBUF$_G$$_{\textit{selectI/O}}$ IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Architectures</th>
<th>Attribute Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Virtex-II</td>
</tr>
<tr>
<td>SSTL3$_{II}$</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>SSTL3$_{II}$/DCI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.

Usage

This design element is supported for schematic and instantiation. Synthesis tools usually infer a BUFGP on any clock net. If there are more clock nets than BUFGPs, the synthesis tool usually instantiates BUFGPs for the clocks that are most utilized. The BUFGP contains both a BUFG and an IBUF.$G$$_{\textit{selectI/O}}$.

VHDL Instantiation Template

```vhdl
-- Component Declaration for IBUF$G$ should be placed
-- after architecture statement but before begin keyword

component IBUF$G$
    port (O : out STD_ULOGIC;
        I : in STD_ULOGIC);
end component;

-- Component Attribute specification for IBUF$G$
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IBUF$G$ should be placed
-- in architecture after the begin keyword

IBUF$G$_INSTANCE_NAME : IBUF$G$
    port map (O => user_O,
        I => user_I);
```

Verilog Instantiation Template

```verilog
IBUF$G$ instance_name (.O (user_O),
                      .I (user_I));
```

Commonly Used Constraints

IOSTANDARD
IOBDELAY
IBUFGDS is a dedicated differential signaling input buffer for connection to the clock buffer (BUFG) or DCM. In IBUFGDS, a design level interface signal is represented as two distinct ports (I and IB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET and MYNETB).

### Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>IBUFGDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The IOSTANDARD attribute values listed in the following table can be applied to an IBUFGDS component to provide SelectIO interface capability. See the Xilinx Constraints Guide for information about using these attributes.

A separate SelectIO component is not provided. Attach an IOSTANDARD attribute to an IBUFGDS and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the inputs for the I/O standard associated with that value.

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Input</th>
<th>VREF Input *</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLVDS_25</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LDT_25</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LDT_25_DT</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LVDS_25 (default)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LVDS_25_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Split</td>
<td>No</td>
<td>2.5</td>
</tr>
</tbody>
</table>
**Usage**

For HDL, this design element is supported for instantiation but not for inference.

**VHDL Instantiation Template**

```
-- Component Declaration for IBUFGDS should be placed
-- after architecture statement but before begin keyword

component IBUFGDS
  port (O : out STD_ULOGIC;
        I : in STD_ULOGIC;
        IB : in STD_ULOGIC);
end component;

-- Component Attribute specification for IBUFGDS
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IBUFGDS should be placed
-- in architecture after the begin keyword

IBUFGDS_INSTANCE_NAME : IBUFGDS
  port map (O => user_O,
            I => user_I,
            IB => user_IB);
```

**Verilog Instantiation Template**

```
IBUFGDS instance_name (.O (user_O),
                      .I (user_I),
                      .IB (user_IB));
```
Commonly Used Constraints

IOSTANDARD
IOBDELAY
IBUFGDS_DIFF_OUT

Differential I/O Input Buffer with Differential Outputs

Architectures Supported

<table>
<thead>
<tr>
<th>IBUFGDS_DIFF_OUT</th>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

IBUFGDS_DIFF_OUT is a differential I/O input buffer with differential outputs. The differential output pair (O & OB) maintains the relation of its differential input pair. Thus, IBUFGDS_DIFF_OUT can be used to maintain the duty cycle of a clock input (even at high frequencies). In order to support this behavior, the outputs of IBUFGDS_DIFF_OUT need to drive a set of matched resources. Our recommended method for achieving this is to use the IBUFGDS_DIFF_OUT primitive driving two BUFGs which, by construction, will use two legs of the global clock network as the matched routing network. An example application is available at XAPP622 on the Xilinx support website at http://www.xilinx.com/xapp/xapp622.pdf.

Usage

IBUFGDS_DIFF_OUT is instantiated rather than inferred.

VHDL Instantiation Template

component IBUFGDS_DIFF_OUT
    port (O : out STD_LOGIC;
          OB : out STD_LOGIC;
          I : in STD_LOGIC;
          IB : in STD_LOGIC);
end component;

Verilog Instantiation Template

module IBUFGDS_DIFF_OUT (O, OB, I, IB);
    output O, OB;
    input I, IB;
endmodule

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Input</th>
<th>VREF Input *</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLVDS_25</td>
<td></td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LDT_25</td>
<td></td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LDT_25_DT</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Commonly Used Constraints

Location constraint: Same as "IBUFGDS"

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Input</th>
<th>VREF Input *</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVDS_25 (default)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LVDS_25_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Split</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDS_25_DT</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDS_33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDS_EXT_25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDS_EXT_25_DCI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDS_EXT_33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDS_EXT_25_DT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVPECL_25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVPECL_33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ULVDS_25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ULVDS_25_DT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.
ICAP_VIRTEX2

User Interface to Virtex-II, Virtex-II Pro, and Virtex-II Pro X Internal Configuration Access Port

Architectures Supported

<table>
<thead>
<tr>
<th>ICAP_VIRTEX, ICAP_VIRTEX2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ICAP_VIRTEX2 provides user access to the Virtex-II, Virtex-II Pro, and Virtex-II Pro X internal configuration access port (ICAP).

Usage

For HDL, this design element is instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for ICAP_VIRTEX2 should be placed
-- after architecture statement but before begin keyword

component ICAP_VIRTEX2
    port (BUSY : out STD_ULOGIC;
          O : out STD_LOGIC_VECTOR (7 downto 0);
          CE : in STD_ULOGIC;
          CLK : in STD_ULOGIC;
          I : in STD_LOGIC_VECTOR (7 downto 0);
          WRITE : in STD_ULOGIC);
end component;

-- Component Attribute specification for ICAP_VIRTEX2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for ICAP_VIRTEX2 should be placed
-- in architecture after the begin keyword

ICAP_VIRTEX2_INSTANCE_NAME : ICAP_VIRTEX2
    port map (BUSY => user_BUSY,
              O => user_O,
              CE => user_CE,
              CLK => user_CLK);
```

I => user_I,
WRITE => user_WRITE);

Verilog Instantiation Template

ICAP_VIRTEX2 instance_name (.BUSY (user_BUSY),
.O (user_O),
.CE (user_CE),
.CLK (user_CLK),
.I (user_I),
.WRITE (user_WRITE));

Commonly Used Constraints

None
IFD, 4, 8, 16

Single- and Multiple-Input D Flip-Flops

Architectures Supported

<table>
<thead>
<tr>
<th>IFD, IFD4, IFD8, IFD16</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The IFD D-type flip-flop is contained in an input/output block (IOB), except for XC9500/XV/XL, CoolRunner XPLA3. The input (D) of the flip-flop is connected to an IPAD or an IOPAD (without using an IBUF). The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin.

The flip-flops are asynchronously cleared with Low outputs when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.
GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

IFD Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

IFD Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

IFD Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFD, you would infer an FD and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
IFD_1

Input D Flip-Flop with Inverted Clock

Architectures Supported

<table>
<thead>
<tr>
<th>IFD_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The IFD_1 D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input also provides data input for the flip-flop, which synchronizes data entering the chip. The D input data is loaded into the flip-flop during the High-to-Low clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin.

The flip-flop is asynchronously cleared with Low output when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
<tr>
<td></td>
<td>Q</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

IFD_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
IFD_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFD_1, you would infer an FD_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
IFDDRCPE

Dual Data Rate Input D Flip-Flop with Clock Enable and Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>IFDDRCPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

IFDDRCPE is a dual data rate (DDR) input D flip-flop with clock enable (CE) and asynchronous preset (PRE) and clear (CLR). It consists of one input buffer and two identical flip-flops (FDCPE).

When the asynchronous PRE is High and CLR is Low, both the Q0 and Q1 outputs are set High. When CLR is High, both outputs are reset Low. When PRE and CLR are Low and CE is High, data on the D input is loaded into the Q0 output on the Low-to-High C0 clock transition, and into the Q1 output on the Low-to-High C1 clock transition.

The flip-flops are asynchronously cleared with Low outputs when power is applied. The INIT attribute does not apply to IFDDRCPE components.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>C1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
</tbody>
</table>
IFDDRCPE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDDRCPE, you would infer an FDDRCPE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.

VHDL Instantiation Template

-- Component Declaration for IFDDRCPE should be placed
-- after architecture statement but before begin keyword

component IFDDRCPE

    -- synthesis translate_on
    port (Q0 : out STD_ULOGIC;
         Q1 : out STD_ULOGIC;
         C0 : in STD_ULOGIC;
         C1 : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         CLR : in STD_ULOGIC;
         D : in STD_ULOGIC;
         PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for IFDDRCPE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IFDDRCPE should be placed
-- in architecture after the begin keyword

IFDDRCPE_INSTANCE_NAME : IFDDRCPE
    port map (Q0 => user_Q0,
              Q1 => user_Q1,
              C0 => user_C0,
IFDDRCPE

C1 => user_C1
CE => user_CE,
CLR => user_CLR,
D => user_D0,
PRE => user_PRE);

Verilog Instantiation Template

IFDDRCPE IFDDRCPE_instance_name (.Q0 (user_Q0),
 .Q1 (user_Q1),
 .C0 (user_C0),
 .C1 (user_C1),
 .CE (user_CE),
 .CLR (user_CLR),
 .D (user_D),
 .PRE (user_PRE));

Commonly Used Constraints

LOC, RLOC, and INIT
IFDDRRSE

Dual Data Rate Input D Flip-Flop with Synchronous Reset and Set and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>IFDDRRSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

IFDDRRSE is a dual data rate (DDR) input D flip-flop with synchronous reset (R), synchronous set (S), and clock enable (CE). It consists of one input buffer and two identical flip-flops (FDRSE).

For the C0 input and Q0 output, reset (R) has precedence. The R input, when High, resets the Q0 output Low during the Low-to-High C0 clock transition. When S is High and R is Low, the Q0 output is set High during the Low-to-High C0 clock transition.

For the C1 input and Q1 output, set (S) has precedence. The R input, when High, resets the Q1 output Low during the Low-to-High C1 clock transition. When S is High and R is Low, the Q0 output is set to High during the Low-to-High C1 clock transition.

The flip-flop is asynchronously cleared, output Low, when power is applied.

The INIT attribute does not apply to IFDDRRSE components.
**Usage**

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDDRRSE, you would infer an FDDRRSE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for IFDDRRSE should be placed
-- after architecture statement but before begin keyword

component IFDDRRSE
    port (Q0 : out STD_ULOGIC;
          Q1 : out STD_ULOGIC;
          C0 : in STD_ULOGIC;
          C1 : in STD_ULOGIC;
          CE : in STD_ULOGIC;
          D : in STD_ULOGIC;
          R : in STD_ULOGIC;
          S : in STD_ULOGIC);
end component;

-- Component Attribute specification for IFDDRRSE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IFDDRRSE should be placed
-- in architecture after the begin keyword

IFDDRRSE_INSTANCE_NAME : IFDDRRSE
    port map (Q0 => user_Q0,
              Q1 => user_Q1,
              C0 => user_C0,
              C1 => user_C1,
              CE => user_CE,
```

In the diagram, the IFDDRRSE component is shown with its inputs and outputs, indicating how it connects to other components in a design. The VHDL code provides a template for declaring, specifying attributes, and instantiating the IFDDRRSE component within a design.
D => user_D,
R => user_R,
S => user_S);

Verilog Instantiation Template

IFDDRRSE IFDDRRSE_instance_name (.Q0 (user_Q0),
    .Q1 (user_Q1),
    .C0 (user_C0),
    .C1 (user_C1),
    .CE (user_CE),
    .D (user_D),
    .R (user_R),
    .S (user_S));

Commonly Used Constraints

LOC
RLOC
INIT
**IFDI**

**Input D Flip-Flop (Asynchronous Preset)**

**Architectures Supported**

<table>
<thead>
<tr>
<th>IFDI</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The IFDI D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin.

The flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

IFDI Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
IFDI Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDI, you would infer an FDP and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
IFDI_1

Input D Flip-Flop with Inverted Clock (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>IFDI_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The IFDI_1 D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the High-to-Low clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin.

The flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

Q 0 1

IFDI_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
**IFDI_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X**

**Usage**

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDI_1, you would infer an FDP_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
**IFDX, 4, 8, 16**

Single- and Multiple-Input D Flip-Flops with Clock Enable

**Architectures Supported**

<table>
<thead>
<tr>
<th>IFDX</th>
<th>IFDX4, IFDX8, IFDX16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The IFDX D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD (without using an IBUF). The D input provides data input for the flip-flop, which synchronizes data entering the chip. When CE is High, the data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin. When CE is Low, flip-flop outputs do not change.

The flip-flops are asynchronously cleared with Low outputs when power is applied. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>Dn</td>
</tr>
<tr>
<td>1</td>
<td>Dn</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>
Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDX, you would infer an FDCE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.

VHDL Instantiation Template

-- Component Declaration for IFDX should be placed
-- after architecture statement but before begin keyword

component IFDX
    port (Q : out STD_ULOGIC;
         C : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         D : in STD_ULOGIC);
end component;

-- Component Attribute specification for IFDX
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IFDX should be placed
-- in architecture after the begin keyword

IFDX_INSTANCE_NAME : IFDX
    port map (Q => user_Q,
              C => user_C,
              CE => user_CE,
              D => user_D);

Verilog Instantiation Template

IFDX IFDX_instance_name (.Q (user_Q),
                        .C (user_C),
                        .CE (user_CE),
                        .D (user_D));

Commonly Used Constraints

IOB
IFDX_1

Input D Flip-Flop with Inverted Clock and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>IFDX_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

The IFDX_1 D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input also provides data input for the flip-flop, which synchronizes data entering the chip. When CE is High, the data on input D is loaded into the flip-flop during the High-to-Low clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin. When the CE pin is Low, the output (Q) does not change.

The flip-flop is asynchronously cleared with Low output, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For more information on IFDX_1, see “ILDX, 4, 8, 16”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

IFDX_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
IFDX_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common
design practice is to infer a regular component and put an IOB=TRUE attribute on the
component in the UCF file or in the code. For instance, to get an IFDX_1, you would
infer an FDCE_1 and put the IOB = TRUE attribute on the component. Or, you could
use the map option –pr i to pack all input registers into the IOBs.
IFDXI

Input D Flip-Flop with Clock Enable (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>IFDXI</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

The IFDXI D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input provides data input for the flip-flop, which synchronizes data entering the chip. When CE is High, the data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin. When the CE pin is Low, the output (Q) does not change.

The flip-flop is asynchronously preset with High output, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For information on legal IFDXI, IFDXI_1, ILDXI, and ILDXI_1 combinations, see “ILDXI”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

IFDXI Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
IFDXI Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDXI, you would infer an FDPE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
**IFDXI_1**

Input D Flip-Flop with Inverted Clock and Clock Enable (Asynchronous Preset)

### Architectures Supported

<table>
<thead>
<tr>
<th>IFDXI_1</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

The IFDXI_1 D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD. The D input provides data input for the flip-flop, which synchronizes data entering the chip. When CE is High, the data on input D is loaded into the flip-flop during the High-to-Low clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin. When the CE pin is Low, the output (Q) does not change.

The flip-flop is asynchronously preset with High output when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For information on legal IFDXI, IFDXI_1, ILDXI, and ILDXI_1 combinations, see "ILDXI".

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

IFDXI_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
IFDXI_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an IFDXI_1, you would infer an FDPE_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
ILD, 4, 8, 16

Transparent Input Data Latches

Architectures Supported

<table>
<thead>
<tr>
<th>architectures</th>
<th>ILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>architectures</th>
<th>ILD4, ILD8, ILD16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ILD, ILD4, ILD8, and ILD16 are single or multiple transparent data latches, which can be used to hold transient data entering a chip. The ILD latch is contained in an input/output block (IOB), except for XC9500/XV/XL. The latch input (D) is connected to an IPAD or an IOPAD (without using an IBUF). When the gate input (G) is High, data on the inputs (D) appears on the outputs (Q). Data on the D inputs during the High-to-Low G transition is stored in the latch.

The latch is asynchronously cleared with Low output when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↓</td>
<td>D</td>
</tr>
</tbody>
</table>
ILD Implementation Spartan-II, Spartan-II E, Virtex, Virtex-E

ILD Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

ILD Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILD, you would infer an LD and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
VHDL Instantiation Template

-- Component Declaration for ILD should be placed
-- after architecture statement but before begin keyword

component ILD
   port (Q : out STD_ULOGIC;
        D : in STD_ULOGIC;
        DG : in STD_ULOGIC);
end component;

-- Component Attribute specification for ILD
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for ILD should be placed
-- in architecture after the begin keyword

ILD_INSTANCE_NAME : ILD
   port map (Q => user_Q,
            D => user_D,
            G => user_G);

Verilog Instantiation Template

ILD ILD_instance_name (.Q (user_Q),
                      .D (user_D),
                      .G (user_G));

Commonly Used Constraints

INIT
ILD_1

Transparent Input Data Latch with Inverted Gate

Architectures Supported

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ILD_1</td>
<td></td>
</tr>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ILD_1 is a transparent data latch, which can be used to hold transient data entering a chip. When the gate input (G) is Low, data on the data input (D) appears on the data output (Q). Data on D during the Low-to-High G transition is stored in the latch.

The latch is asynchronously cleared with Low output when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>D</td>
</tr>
</tbody>
</table>

ILD_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
**ILD_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X**

**Usage**

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILD_1, you would infer an LD_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for ILD_1 should be placed
-- after architecture statement but before begin keyword

component ILD_1
    port (Q : out STD_ULOGIC;
         D : in STD_ULOGIC;
         G : in STD_ULOGIC);
end component;

-- Component Attribute specification for ILD_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for ILD_1 should be placed
-- in architecture after the begin keyword

ILD_1_INSTANCE_NAME : ILD_1
    port map (Q => user_Q,
              D => user_D,
              G => user_G);
```

**Verilog Instantiation Template**

```verilog
ILD_1 ILD_1_instance_name (.Q (user_Q),
                          .D (user_D),
                          .G (user_G));
```

**Commonly Used Constraints**

INIT
ILDIs and IFDIs

The ILDI is actually the input flip-flop master latch. It is possible to access two different outputs from the input flip-flop: one that responds to the level of the clock signal and another that responds to an edge of the clock signal. When using both outputs from the same input flip-flop, a transparent High latch (ILDIs) corresponds to a falling edge-triggered flip-flop (IFDI_1). Similarly, a transparent Low latch (ILDIs_1) corresponds to a rising edge-triggered flip-flop (IFDI).
ILD implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILDI, you would infer an LDP and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
ILDI_1

Transparent Input Data Latch with Inverted Gate (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>ILDI_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro*</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

* Macros cannot be implemented for Spartan-IIIE.

ILDI_1 is a transparent data latch, which can hold transient data entering a chip. When the gate input (G) is Low, data on the data input (D) appears on the data output (Q). Data on D during the Low-to-High G transition is stored in the latch.

The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For information on ILDI_1, see “ILDI”.

Inputs | Outputs
-------|-------
| G      | D     | Q     |
| 0      | 1     | 1     |
| 0      | 0     | 0     |
| 1      | X     | D     |
| ↑      | D     | D     |
ILDI_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILDI_1, you would infer an LDP_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
ILDX, 4, 8, 16

Transparent Input Data Latches

Architectures Supported

<table>
<thead>
<tr>
<th>ILDX</th>
<th>Architectures Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ILDX4, ILDX8, ILDX16</th>
<th>Architectures Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ILDX, ILDX4, ILDX8, and ILDX16 are single or multiple transparent data latches, which can be used to hold transient data entering a chip. The latch input (D) is connected to an IPAD or an IOPAD (without using an IBUF).

The latch is asynchronously cleared, output Low, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

ILDXs and IFDXs

The ILDX is actually the input flip-flop master latch. Two different outputs can be accessed from the input flip-flop: one that responds to the level of the clock signal and another that responds to an edge of the clock signal. When using both outputs from the same input flip-flop, a transparent High latch (ILDX) corresponds to a falling edge-triggered flip-flop (IFDX_1). Similarly, a transparent Low latch (ILDX_1) corresponds to a rising edge-triggered flip-flop (IFDX).
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>G</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

ILDX Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

ILDX Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
ILDX8 Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILDX, you would infer an LDCE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
ILDX_1

Transparent Input Data Latch with Inverted Gate

Architectures Supported

<table>
<thead>
<tr>
<th>ILDX_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ILDX_1 is a transparent data latch, which can be used to hold transient data entering a chip. When the gate input (G) is Low, data on the data input (D) appears on the data output (Q). Data on D during the Low-to-High G transition is stored in the latch.

The latch is asynchronously cleared with Low output, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For more information on ILDX_1, see “ILDX, 4, 8, 16”.

<table>
<thead>
<tr>
<th></th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>G</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>D</td>
</tr>
</tbody>
</table>

ILDX_1 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
ILDX_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILDX_1, you would infer an LDCE_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
ILDXI

Transparent Input Data Latch (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>ILDXI</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ILDXI is a transparent data latch, which can hold transient data entering a chip. When the gate input (G) is High, data on the input (D) appears on the output (Q). Data on the D input during the High-to-Low G transition is stored in the latch.

The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

ILDXIs and IFDXIs

The ILDXI is actually the input flip-flop master latch. Two different outputs can be accessed from the input flip-flop: one that responds to the level of the clock signal and another that responds to an edge of the clock signal. When using both outputs from the same input flip-flop, a transparent High latch (ILDXI) corresponds to a falling edge-triggered flip-flop (IFDXI_1). Similarly, a transparent Low latch (ILDXI_1) corresponds to a rising edge-triggered flip-flop (IFDXI). See the following figure for legal IFDXI, IFDXI_1, ILDXI, and ILDXI_1 combinations.

Legal Combinations of IFDXI and ILDXI for a Single IOB
### Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILDXI, you would infer an LDPE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
ILDXI_1

Transparent Input Data Latch with Inverted Gate (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>ILDXI_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

ILDXI_1 is a transparent data latch, which can hold transient data entering a chip. The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

For information on legal IFDXI, IFDXI_1, ILDXI, and ILDXI_1 combinations, see “ILDXI”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>G</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

ILDXI_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
ILDXI_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an ILDXI_1, you would infer an LDPE_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr i to pack all input registers into the IOBs.
Single and Multiple Inverters

Architectures Supported

### INV

<table>
<thead>
<tr>
<th>Spartan-II, Spartan-IIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

### INV4, INV8, INV16

<table>
<thead>
<tr>
<th>Spartan-II, Spartan-IIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

INV, INV4, INV8, and INV16 are single and multiple inverters that identify signal inversions in a schematic.

Usage

For HDL, this design element can be instantiated or inferred.
VHDL Inference Code

architecture Behavioral of inv is

begin

process (i)
begin
  o <= not i;
end process;

end Behavioral

Verilog Inference Code

always @(i)
begin
  o = !i;
end

VHDL Instantiation Template

-- Component Declaration for INV should be placed
-- after architecture statement but before begin keyword

component INV
  port (O : out STD_ULOGIC;
       I : in STD_ULOGIC);
end component;

-- Component Attribute specification for INV
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for INV should be placed
-- in architecture after the begin keyword

INV_INSTANCE_NAME : INV
  port map (O => user_O,
            I => user_I);

Verilog Instantiation Template

INV instance_name (.O (user_O),
                   .I (user_I));

Commonly Used Constraints

None
IOBUF, IOBUF_selectIO

Bi-Directional Buffer with Selectable I/O Interface

Architectures Supported

<table>
<thead>
<tr>
<th>IOBUF, IOBUF_selectIO</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

For Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, IOBUF and its selectIO variants (listed in the "Component" column in the table below) are bi-directional buffers whose I/O interface corresponds to a specific I/O standard. The name extensions (LVCMOS2, PCI33_3, PCI33_5, etc.) specify the standard. The S, F, and 2, 4, 6, 8, 12, 16, 24 extensions specify the slew rate (SLOW or FAST) and the drive power (2, 4, 6, 8, 12, 16, 24 mA) for the LVTTL standard variants. For example, IOBUF_F_2 is a bi-directional buffer that uses the LVTTL I/O-signaling standard with a FAST slew and 2mA of drive power. You can attach an IOSTANDARD attribute to an IOBUF instance instead of using an IOBUF_selectIO component. Check marks (√) in the “Spartan-II, Virtex” and “Spartan-IIE, Virtex-E” columns indicate the components and IOSTANDARD attribute values available for each architecture.

IOBUF components that use the LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33 signaling standards have selectable drive and slew rates using the DRIVE and FAST or SLOW constraints. The defaults are DRIVE=12 mA and SLOW slew.

IOBUFs are composites of IBUF and OBUFT elements. The O output is X (unknown) when IO (input/output) is Z. IOBUFs can be implemented as interconnections of their component elements.

The hardware implementation of the I/O standards requires that you follow a set of usage rules for the SelectIO buffers. See the “SelectIO Usage Rules” under the IBUF_selectIO section for information on using these components and IOSTANDARD attributes.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Bidirectional</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>I</td>
<td>IO</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Spartan-II, Spartan-IIIE, Virtex, and Virtex-E IOBUF\_selectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-IIIE, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>VREF</th>
<th>Output VCCO</th>
<th>Input VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBUF</td>
<td>√</td>
<td>√</td>
<td>defaults to LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_2</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_4</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_6</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_8</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_12</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_16</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_S_24</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_2</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_4</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_6</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_8</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_12</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_16</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_F_24</td>
<td>√</td>
<td>√</td>
<td>LVTTL(^b)</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_AGP</td>
<td>√</td>
<td>√</td>
<td>AGP</td>
<td>1.32</td>
<td>3.3</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_CTT</td>
<td>√</td>
<td>√</td>
<td>CTT</td>
<td>1.50</td>
<td>3.3</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_GTL</td>
<td>√</td>
<td>√</td>
<td>GTL</td>
<td>0.80</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_GTP</td>
<td>√</td>
<td>√</td>
<td>GTP</td>
<td>0.80</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_HSTL_I</td>
<td>√</td>
<td>√</td>
<td>HSTL_I</td>
<td>0.75</td>
<td>1.5</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_HSTL_III</td>
<td>√</td>
<td>√</td>
<td>HSTL_III</td>
<td>0.90</td>
<td>1.5</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_HSTL_IV</td>
<td>√</td>
<td>√</td>
<td>HSTL_IV</td>
<td>0.90</td>
<td>1.5</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_LVCMOS2</td>
<td>√</td>
<td>√</td>
<td>LVCMOS2</td>
<td>No</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>IOBUF_LVCMOS18</td>
<td>√</td>
<td>√</td>
<td>LVCMOS18(^b)</td>
<td>No</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>IOBUF_LVDS</td>
<td>√</td>
<td>√</td>
<td>LVDS</td>
<td>No</td>
<td>2.5</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_LVPECL</td>
<td>√</td>
<td>√</td>
<td>LVPECL</td>
<td>No</td>
<td>3.3</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_PCI33_3</td>
<td>√</td>
<td>√</td>
<td>PCI33_3</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_PCI33_5</td>
<td>√</td>
<td>√(^a)</td>
<td>PCI33_5</td>
<td>No</td>
<td>3.3</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_PCI66_3</td>
<td>√</td>
<td>√</td>
<td>PCI66_3</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_PCI_X66_3</td>
<td>√</td>
<td>√</td>
<td>PCI_X66_3</td>
<td>No</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>IOBUF_SSTL2_I</td>
<td>√</td>
<td>√</td>
<td>SSTL2_I</td>
<td>1.25</td>
<td>2.5</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_SSTL2_II</td>
<td>√</td>
<td>√</td>
<td>SSTL2_II</td>
<td>1.25</td>
<td>2.5</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_SSTL3_I</td>
<td>√</td>
<td>√</td>
<td>SSTL3_I</td>
<td>1.50</td>
<td>3.3</td>
<td>No</td>
</tr>
<tr>
<td>IOBUF_SSTL3_II</td>
<td>√</td>
<td>√</td>
<td>SSTL3_II</td>
<td>1.50</td>
<td>3.3</td>
<td>No</td>
</tr>
</tbody>
</table>

\(^a\) Not supported for Virtex-E.

\(^b\) The LVCMOS18 attribute also requires a slew value (FAST or SLOW) and DRIVE value. See the FAST, SLOW, and DRIVE attribute descriptions in the Xilinx Constraints Guide for valid values for each architecture.
The Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X library includes some IOBUF_selectIO components for compatibility with older, existing designs and other architectures. For new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs, however, the recommended method for using IOBUF SelectIO buffers is to attach an IOSTANDARD attribute to an IOBUF component. For example, attach IOSTANDARD=GTLP to an IOBUF instead of using the IOBUF_GTLP component for new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs. The IOSTANDARD attributes that can be attached to an IOBUF component are listed in the "IOSTANDARD (Attribute Value)" column in the following table “Spartan-II, Spartan-IIIE, Virtex, and Virtex-E IOBUF selectIO Components and IOSTANDARD Attributes” table. See “SelectIO Usage Rules” for information on using these IOSTANDARD attributes.

For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the IOSTANDARD attribute values listed in the following table can be applied to an IOBUF component to provide SelectIO interface capability for the inputs. The O output uses the LVTTL standard.

Attach an IOSTANDARD attribute to an IOBUF and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the input for the I/O standard associated with that value.

The LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33 attributes also require a slew value (FAST or SLOW) and DRIVE value. See the FAST, SLOW, and DRIVE attribute descriptions in the Xilinx Constraints Guide for valid values for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X.

### Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X IOBUF SelectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Architectures</th>
<th>Attribute Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Virtex-II</td>
</tr>
<tr>
<td>AGP</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>GTL</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>GTL_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>GTLP</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_II_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_II_DCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_III</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_III_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>HSTL_IV_DCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVCMOS12a</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>LVCMOS15a</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>LVCMOS18a</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>LVCMOS25a</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>LVCMOS33a</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>IOSTANDARD</td>
<td>Spartan-3</td>
<td>Virtex-II</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>LVDCI_15</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_25</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_33</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_15</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_18</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_25</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVDCI_DV2_33</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>LVTTL (default)*</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PCI33_3</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PCI66_3</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PCIX</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL18_II_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_II_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL3_II_DCI</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL3_I</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SSTL3_II</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.

### Usage

For HDL, these design elements are instantiated rather than inferred.

### VHDL Instantiation Template for IOBUF

```vhdl
-- Component Declaration for IOBUF should be placed
-- after architecture statement but before begin keyword

component IOBUF
   port (O : out STD_ULOGIC;
         IO : inout STD_ULOGIC;
         I : in STD_ULOGIC;
         T : in STD_ULOGIC);
end component;

-- Component Attribute specification for IOBUF
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IOBUF should be placed
-- in architecture after the begin keyword
```

1098 www.xilinx.com
1-800-255-7778
Libraries Guide
ISE 6.i
IOBUF, IOBUF_selectIO

**Verilog Instantiation Template for IOBUF**

IOBUF instance_name (.O (user_O),
    .IO (user_IO),
    .I (user_I),
    .T (user_T));

**VHDL Instantiation Template for IOBUF_selectIO**

-- Component Declaration for IOBUF_selectIO should be placed
-- after architecture statement but before begin keyword

component IOBUF_selectIO
    port (O : out STD_ULOGIC;
         IO : inout STD_ULOGIC;
         I : in STD_ULOGIC;
         T : in STD_ULOGIC);
end component;

-- Component Attribute specification for IOBUF_selectIO
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IOBUF_selectIO should be
-- placed in architecture after the begin keyword

IOBUF_selectIO_INSTANCE_NAME : IOBUF_selectIO
    port map (O => user_O,
              IO => user_IO,
              I => user_I,
              T => user_T);

**Verilog Instantiation Template for IOBUF_selectIO**

IOBUF_selectIO instance_name (.O (user_O),
    .IO (user_IO),
    .I (user_I),
    .T (user_T));

**Commonly Used Constraints**

IOSTANDARD, DRIVE, and SLEW
IOBUFDS

3-State Differential Signaling I/O Buffer with Active Low Output Enable

Architectures Supported

<table>
<thead>
<tr>
<th>IOBUFDS</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
<th>XC9500, XC9500XV, XC9500XL</th>
<th>CoolRunner XPLA3</th>
<th>CoolRunner-II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

IOBUFDS is a single 3-state, differential signaling input/output buffer with active Low output enable.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Bidirectional</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>T</td>
<td>IO</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

* The dash (-) means No Change.

BLVDS_25 is supported for IOSTANDARD. Attach an IOSTANDARD attribute to an IOBUFDS and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the outputs for the I/O standard associated with that value.

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLVDS_25</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>

VREF  Input

Output VCCO

Input VCCO

* VREF requirement when this IOSTANDARD is an input.

Usage

For HDL, this design element is instantiated rather than inferred.
VHDL Instantiation Template

-- Component Declaration for IOBUFDS should be placed
-- after architecture statement but before begin keyword

component IOBUFDS
    port (O : out STD_ULOGIC;
          IO : inout STD_ULOGIC;
          IOB : inout STD_ULOGIC;
          I : in STD_ULOGIC;
          T : in STD_ULOGIC);
end component;

-- Component Attribute specification for IOBUFDS
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for IOBUFDS should be
-- placed in architecture after the begin keyword

IOBUFDS_INSTANCE_NAME : IOBUFDS
    port map (O => user_O,
              IO => user_IO,
              IOB => user_IOB,
              I => user_I,
              T => user_T);

Verilog Instantiation Template

IOBUFDS instance_name (.O (user_O),
                       .IO (user_IO),
                       .IOB (user_IOB),
                       .I (user_I),
                       .T (user_T));

Commonly Used Constraints

IOSTANDARD
IOBDELAY
IOPAD, 4, 8, 16

Single- and Multiple-Input/Output Pads

<table>
<thead>
<tr>
<th>IOPAD</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOPAD4, IOPAD8, IOPAD16</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

IOPAD, IOPAD4, IOPAD8, and IOPAD16 are single and multiple input/output pads. The IOPAD is a connection point from a device pin, used as a bidirectional signal, to a PLD device. The IOPAD is connected internally to an input/output block (IOB), which is configured by the software as a bidirectional block. Bidirectional blocks can consist of any combination of a 3-state output buffer (such as OBUFT or OFDE) and any available input buffer (such as IBUF or IFD). See the appropriate CAE tool interface user guide for details on assigning pin location and identification.

**Note:** The LOC attribute cannot be used on IOPAD multiples.

Usage
For HDL, it is not necessary to use these elements in the design. They will be added automatically.

Commonly Used Constraints
IOBDELAY
PULLDOWN
IPAD, IPAD4, IPAD8, and IPAD16 are single and multiple input pads. The IPAD is a connection point from a device pin used for an input signal to the PLD device. It is connected internally to an input/output block (IOB), which is configured by the software as an IBUF, IFD, or ILD. See the appropriate CAE tool interface user guide for details on assigning pin location and identification.

For Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, pads must be used to drive IBUF and IBUFG inputs. An IPAD can be inferred by NGDBUILD if one is missing on an IBUF or IBUFG input. **Note:** The LOC attribute cannot be used on IPAD multiples.

Usage

For HDL, it is not necessary to use these elements in the design. They will be added automatically.

Commonly Used Constraints

IOBDELAY, PULLDOWN
JTAGPPC

JTAG Primitive for the Power PC

Architectures Supported

<table>
<thead>
<tr>
<th>JTAGPPC</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive*</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

* Not supported for Virtex-II. Supported for Virtex-II Pro and Virtex-II Pro X only.

The JTAGPPC block allows connection from the JTAG logic in the PPC405 core to the JTAG logic of Virtex-II Pro and Virtex-II Pro X devices. The connections are made through programmable routing and so the connection only exists after configuration. Following is an example instantiation of the JTAGPPC block in Verilog:

```verilog
JTAGPPC JTAGPPC(.TDOTSPPC(TDO_TS_PPC),
                 .TDOPPC(TDO_PPC), .TMS(TMS_PPC),
                 .TDIPPC(TDI_PPC), .TCK(TCK_PPC));

PPC405 IPPC405 (
    .JTGC405TCK (TCK_PPC),
    .JTGC405TDI (TDI_PPC),
    .JTGC405TMS (TMS_PPC),
    .C405JTGTDO (TDO_PPC),
    .C405JTGTDOEN (TDO_TS_PPC),
    ... )
```

When the block is instantiated in this fashion, the instruction registers of the PPC405 and the Virtex-II Pro and Virtex-II Pro X devices are linked in series.

The following table lists the input and output pins for JTAGPPC.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDOPPC</td>
<td>TCK</td>
</tr>
<tr>
<td>TDOTSPPC</td>
<td>TDIPPC</td>
</tr>
<tr>
<td></td>
<td>TMS</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is instantiated rather than inferred.
VHDL Instantiation Template

-- Component Declaration for JTAGPPC should be placed
-- after architecture statement but before begin keyword

component JTAGPPC
    port (TCK : out STD_ULOGIC;
        TDIPPC : out STD_ULOGIC;
        TMS : out STD_ULOGIC;
        TDOPPC : in STD_ULOGIC;
        TDOTSPPC : in STD_ULOGIC);
end component;

-- Component Attribute specification for JTAGPPC
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for JTAGPPC should be placed
-- in architecture after the begin keyword

JTAGPPC_INSTANCE_NAME : JTAGPPC
    port map (TCK => user_TCK,
        TDIPPC => user_TDIPPC,
        TMS => user_TMS,
        TDOPPC => user_TDOPPC,
        TDOTSPPC => user_TDOTSPPC);

Verilog Instantiation Template

JTAGPPC JTAGPPC_instance_name (.TCK (user_TCK),
    .TDIPPC (user_TDIPPC),
    .TMS (user_TMS),
    .TDOPPC (user_TDOPPC),
    .TDOTSPPC (user_TDOTSPPC));

Commonly Used Constraints

None
KEEPER

KEEPER Symbol

Architectures Supported

<table>
<thead>
<tr>
<th>KEEPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>* Primitive for XC9500XL/XV</td>
</tr>
</tbody>
</table>

KEEPER is a weak keeper element used to retain the value of the net connected to its bidirectional O pin. For example, if a logic 1 is being driven onto the net, KEEPER drives a weak/resistive 1 onto the net. If the net driver is then 3-stated, KEEPER continues to drive a weak/resistive 1 onto the net.

For additional information on using a KEEPER element with SelectIO components, see the “SelectIO Usage Rules” in the "IBUF_selectIO" section

Usage

For HDL, this design element is instantiated rather than inferred.

VHDL Instantiation Template

```
-- Component Declaration for KEEPER should be placed
-- after architecture statement but before begin keyword
component KEEPER
  port (O : inout STD_ULOGIC);
end component;

-- Component Attribute specification for KEEPER
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for KEEPER should be placed
-- in architecture after the begin keyword

KEEPER_INSTANCE_NAME : KEEPER
  port map (O => user_O);
```
Verilog Instantiation Template

KEEPER KEEP_instance_name (.O (user_O));
LD is a transparent data latch. The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is High. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

The latch is asynchronously cleared, output Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP SPARTAN2, STARTUP SPARTAN3, STARTUP VIRTEX, or STARTUP VIRTEX2 symbol.
LD Implementation XC9500/XV/XL

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```
architecture Behavioral of ld is
begin
  process (G,D)
  begin
    if (G = '1') then
      Q <= D;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```
always @(D or G) begin
  if (G)
    Q <= D;
end
```

VHDL Instantiation Template

```
component LD
  -- Component Declaration for LD should be placed
  -- after the architecture declaration but before the begin keyword
  generic (
    INIT : bit := '1');
  port (Q : out STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC);
end component;

-- Component Attribute specification for LD
-- should be placed after the architecture declaration but
-- before the begin keyword
```

attribute INIT : string;
attribute INIT of LD_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LD should be placed
-- in architecture after the begin keyword

LD_INSTANCE_NAME : LD
   -- synthesis translate_off
   generic map (    
       INIT => bit_value)
   -- synthesis translate_on
   port map (Q => user_Q, 
            D => user_D, 
            G => user_G);

Verilog Instantiation Template
LD LD_instance_name (.Q (user_Q),
                 .D (user_D),
                 .G (user_G));

defparam LD_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LD4, LD8, LD16 have, respectively, 4, 8, and 16 transparent data latches with a common gate enable (G). The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is High. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

See “LD” for information on single transparent data latches.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↓</td>
<td>D</td>
</tr>
</tbody>
</table>

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```
arquitectura Behavioral de ld4 es
begin
    proceso (G,D)
        begin
            si (G = '1') entonces
                Q <= D;
            final si;
        end proceso;
    final Behavioral;
```

Verilog Inference Code

```
siempre Q (D o G) comienza
    si (G)
        Q <= D;
    final
```

X9470
LD_1

Transparent Data Latch with Inverted Gate

Architectures Supported

<table>
<thead>
<tr>
<th>LD_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LD_1 is a transparent data latch with an inverted gate. The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>D</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of ld_1 is
begin
process (G, D)
begin
end process
end
```
if (G='0') then
  Q <= D;
end if;
end process;

end Behavioral

Verilog Inference Code

always @(D or G) begin
  if (!G)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LD_1 should be placed
-- after architecture statement but before begin keyword

cOMPONENT LD_1
  -- synthesis translate_off
  GENERIC (
    INIT : bit := '1');
  -- synthesis translate_on
  PORT (Q : OUT STD_ULOGIC;
        D : IN STD_ULOGIC;
        G : IN STD_ULOGIC);
END COMPONENT;

-- Component Attribute specification for LD_1
-- should be placed after architecture declaration but
-- before the begin keyword

ATTRIBUTE INIT : STRING;
ATTRIBUTE INIT OF LD_1_INSTANCE_NAME : Label IS "0";
-- values can be (0 or 1)

-- Component Instantiation for LD_1 should be placed
-- in architecture after the begin keyword

LD_1_INSTANCE_NAME : LD_1
  -- synthesis translate_off
  GENERIC MAP(
    INIT => bit_value);
  -- synthesis translate_on
  PORT MAP (Q => user_Q,
            D => user_D,
            G => user_G);

Verilog Instantiation Template

LD_1 LD_1_INSTANCE_NAME (.Q (user_Q),
                         .D (user_D),
                         .G (user_G));

defparam LD_1_INSTANCE_NAME.INIT = bit_value;
Commonly Used Constraints

INIT
LDC

Transparent Data Latch with Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>LDC</th>
<th></th>
</tr>
</thead>
</table>
| Spartan-II, Spartan-IIE | Primitive
| Spartan-3          | Primitive
| Virtex, Virtex-E    | Primitive
| Virtex-II, Virtex-II Pro, Virtex-II Pro X | Primitive
| XC9500, XC9500XV, XC9500XL | Macro
| CoolRunner XPLA3    | Primitive
| CoolRunner-II       | Primitive

LDC is a transparent data latch with asynchronous clear. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate enable (G) input is High and CLR is Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains low.

The latch is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

```
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
</tbody>
</table>
```

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of ldc is
begin
```
process (CLR, D, G)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (G='1') then
    Q <= D;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (G or D or CLR) begin
  if (CLR)
    Q <= 0;
  else if (G)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDC should be placed
-- after architecture statement but before begin keyword

component LDC
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    CLR : in STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDC
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDC_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDC should be placed
-- in architecture after the begin keyword

LDC_INSTANCE_NAME : LDC
  -- synthesis translate_off
  generic map (
    INIT => bit_value
  )
  -- synthesis translate_on
  port map (Q => user_Q,
    CLR => user_CLR,
    D => user_D,
    G => user_G);
Verilog Instantiation Template

LDC LDC_instance_name (.Q (user_Q),
    .CLR (user_CLR),
    .D (user_D),
    .G (user_G));

defparam LDC_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDC_1

Transparent Data Latch with Asynchronous Clear and Inverted Gate

Architectures Supported

<table>
<thead>
<tr>
<th>LDC_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LDC_1 is a transparent data latch with asynchronous clear and inverted gate. When the asynchronous clear input (CLR) is High, it overrides the other inputs (D and G) and resets the data (Q) output Low. Q reflects the data (D) input while the gate enable (G) input and CLR are Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldc_1 is
begin
    process (CLR, D, G)
    begin
        if (CLR='1') then
            Q <= '0';
        elsif (G='0') then
            Q <= D;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (G or D or CLR) begin
    if (CLR)
        Q <= 0;
    else if (!G)
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDC_1 should be placed
-- after architecture statement but before begin keyword

component LDC_1
    -- synthesis translate_off
    generic (
        INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
    CLR : in STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDC_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDC_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDC_1 should be placed
-- in architecture after the begin keyword

LDC_1_INSTANCE_NAME : LDC_1
    -- synthesis translate_off
    generic map(
        INIT => bit_value
    -- synthesis translate_on
    port map (Q => user_Q,
    CLR => user_CLR,
D => user_D,
G => user_G);

Verilog Instantiation Template

LDC_1 LDC_1_instance_name (.Q (user_Q),
        .CLR (user_CLR),
        .D (user_D),
        .G (user_G));

defparam LDC_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDCE

Transparent Data Latch with Asynchronous Clear and Gate Enable

Architectures Supported

<table>
<thead>
<tr>
<th>LDCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>No</td>
</tr>
</tbody>
</table>

LDCE is a transparent data latch with asynchronous clear and gate enable. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High and CLR is Low. If GE is Low, data on D cannot be latched. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains low.

The latch is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>GE</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>No Chg</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldce is
begin
process (CLR, D, G, GE)
begin
if (CLR='1') then
    Q <= '0';
elsif (GE = '1') then
    if (G='1') then
        Q <= D;
    end if;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @(CLR or D or G or GE) begin
    if (CLR)
        Q <= 0;
    else if (G && GE)
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDCE should be placed
-- after architecture statement but before begin keyword

component LDCE
    -- synthesis translate_off
    generic (INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
        CLR : in STD_ULOGIC;
        D : in STD_ULOGIC;
        G : in STD_ULOGIC;
        GE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDCE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDCE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDCE should be placed
-- in architecture after the begin keyword

LDCE_INSTANCE_NAME : LDCE
-- synthesis translate_off
generic map (  
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,  
  CLR => user_CLR,  
  D => user_D,  
  G => user_G,  
  GE => user_GE);

Verilog Instantiation Template

LDCE  LDCE_instance_name (.Q (user_Q),  
  .CLR (user_CLR),  
  .D (user_D),  
  .G (user_G),  
  .GE (user_GE));

defparam LDCE_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDCE_1

Transparent Data Latch with Asynchronous Clear, Gate Enable, and Inverted Gate

**Architectures Supported**

<table>
<thead>
<tr>
<th>LDCE_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LDCE_1 is a transparent data latch with asynchronous clear, gate enable, and inverted gate. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and CLR are Low and gate enable (GE) is High. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High or GE remains Low.

The latch is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>GE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Usage**

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldce_1 is
begin
process (CLR, D, G, GE)
begin
if (CLR='1') then
  Q <= '0';
elsif (GE = '1') then
  if (G='0') then
    Q <= D;
  end if;
end if;
end process;
end Behavioral;

Verilog Inference Code
always @(CLR or D or G or GE) begin
  if (CLR)
    Q <= 0;
  else if (!G && GE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDCE_1 should be placed
-- after architecture statement but before begin keyword

component LDCE_1
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    CLR : in STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC;
    GE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDCE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDCE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDCE_1 should be placed
-- in architecture after the begin keyword

LDCE_1_INSTANCE_NAME : LDCE_1
-- synthesis translate_off
generic map (
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
  CLR => user_CLR,
  D => user_D,
  G => user_G,
  GE => user_GE);

Verilog Instantiation Template

LDCE_1 LDCE_1_instance_name (.Q (user_Q),
  .CLR (user_CLR),
  .D (user_D),
  .G (user_G),
  .GE (user_GE));

defparam LDCE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LD4CE, LD8CE, LD16CE

Transparent Data Latches with Asynchronous Clear and Gate Enable

Architectures Supported

<table>
<thead>
<tr>
<th>LD4CE, LD8CE, LD16CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LD4CE, LD8CE, and LD16CE have, respectively, 4, 8, and 16 transparent data latches with asynchronous clear and gate enable. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) outputs Low. Q reflects the data (D) inputs while the gate (G) and gate enable (GE) are High, and CLR is Low. If GE is Low, data on D cannot be latched. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains Low.

The latch is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>GE</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Dn = referenced input, for example, D0, D1, D2
Qn = referenced output, for example, Q0, Q1, Q2

LD8CE Implementation Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage
For HDL, these design elements are supported for inference only.

VHDL Inference Code

architecture Behavioral of ld4ce is

begin

process (CLR, D, G, GE)
begin
    if (CLR='1') then
        Q <= "0000";
    elsif (GE = '1') then
        if (G='1') then
            Q <= D;
        end if;
    end if;
end process;
end Behavioral;

Verilog Inference Code

always @(CLR or D or G or GE) begin
    if (CLR)
        Q <= 0;
    else if (G && GE)
        Q <= D;
end
LDCP

Transparent Data Latch with Asynchronous Clear and Preset

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>LDCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

LDCP is a transparent data latch with data (D), asynchronous clear (CLR) and preset (PRE) inputs. When CLR is High, it overrides the other inputs and resets the data (Q) output Low. When PRE is High and CLR is low, it presets the data (Q) output High. Q reflects the data (D) input while the gate (G) input is High and CLR and PRE are Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldcp is
begin
process (CLR, D, G, PRE)
begin
if (CLR='1') then
Q <= '0';
elsif (PRE='1') then
Q <= '1';
elsif (G='1') then
Q <= D;
end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (CLR or PRE or D or G) begin
if (CLR)
    Q <= 0;
else if (PRE)
    Q <= 1;
else if (G)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDCP should be placed
-- after architecture statement but before begin keyword

component LDCP
-- synthesis translate_off
generic ( 
  INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
  CLR : in STD_ULOGIC;
  D : in STD_ULOGIC;
  G : in STD_ULOGIC;
  PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDCP
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDCP_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDCP should be placed
-- in architecture after the begin keyword
LDCP

LDCP_INSTANCE_NAME : LDCP
    -- synthesis translate_off
    generic map (
      INIT => bit_value)
    -- synthesis translate_on
    port map (Q => user_Q,
             CLR => user_CLR,
             D => user_D,
             G => user_G,
             PRE => user_PRE);

Verilog Instantiation Template

LDCP LDCP_instance_name (.Q (user_Q),
                        .CLR (user_CLR),
                        .D (user_D),
                        .G (user_G),
                        .PRE (user_PRE));

defparam LDCP_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDCP_1

Transparent Data Latch with Asynchronous Clear and Preset and Inverted Gate

Architectures Supported

<table>
<thead>
<tr>
<th>LDCP_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LDCP_1 is a transparent data latch with data (D), asynchronous clear (CLR), preset (PRE) inputs, and inverted gate (G). When CLR is High, it overrides the other inputs and resets the data (Q) output Low. When PRE is High and CLR is Low, it presets the data (Q) output High. Q reflects the data (D) input while gate (G) input, CLR, and PRE are Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldcp_1 is
begin
process (CLR, D, G, PRE)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (PRE='1') then
    Q <= '1';
  elsif (G='0') then
    Q <= D;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @(CLR or PRE or D or G) begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (!G)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDCP_1 should be placed
-- after architecture statement but before begin keyword

component LDCP_1
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        CLR : in STD_ULOGIC;
        D : in STD_ULOGIC;
        G : in STD_ULOGIC;
        PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDCP_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDCP_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDCP_1 should be placed
-- in architecture after the begin keyword
LDPC_1_INSTANCE_NAME : LDCP_1
-- synthesis translate_off
  generic map (
      INIT => bit_value)
-- synthesis translate_on
  port map (Q => user_Q,
            CLR => user_CLR,
            D => user_D,
            G => user_G,
            PRE => user_PRE);

Verilog Instantiation Template

LDPC_1 LDCP_1_instance_name (.Q (user_Q),
                              .CLR (user_CLR),
                              .D (user_D),
                              .G (user_G),
                              .PRE (user_PRE));

defparam LDCP_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDCPE

Transparent Data Latch with Asynchronous Clear and Preset and Gate Enable

Architectures Supported

<table>
<thead>
<tr>
<th>LDCPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LDCPE is a transparent data latch with data (D), asynchronous clear (CLR), asynchronous preset (PRE), and gate enable (GE). When CLR is High, it overrides the other inputs and resets the data (Q) output Low. When PRE is High and CLR is Low, it presets the data (Q) output High. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High and CLR and PRE are Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains Low.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>PRE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

```vhdl
architecture Behavioral of ldcpe is
begin
process (CLR, D, G, GE, PRE)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (PRE='1') then
    Q <= '1';
  elsif (GE = '1') then
    if (G='1') then
      Q <= D;
    end if;
  end if;
end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (CLR or PRE or D or G or GE) begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (G && GE)
    Q <= D;
end
```

VHDL Instantiation Template

```vhdl
-- Component Declaration for LDCPE should be placed
-- after architecture statement but before begin keyword
component LDCPE
  -- synthesis translate_off
  generic (INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        CLR : in STD_ULOGIC;
        D : in STD_ULOGIC;
        G : in STD_ULOGIC;
        GE : in STD_ULOGIC;
        PRE : in STD_ULOGIC);
end component;
-- Component Attribute specification for LDCPE
```
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDCPE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDCPE should be placed
-- in architecture after the begin keyword

LDCPE_INSTANCE_NAME : LDCPE
-- synthesis translate_off
generic map (  
    INIT => bit_value)  
-- synthesis translate_on
port map (Q => user_Q,
         CLR => user_CLR,
         D => user_D,
         G => user_G,
         GE => user_GE,
         PRE => user_PRE);

Verilog Instantiation Template

LDCPE LDCPE_instance_name (.Q (user_Q),
                        .CLR (user_CLR),
                        .D (user_D),
                        .G (user_G),
                        .GE (user_GE),
                        .PRE (user_PRE));

defparam LDCPE_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDCPE_1

Transparent Data Latch with Asynchronous Clear and Preset, Gate Enable, and Inverted Gate

**Architectures Supported**

<table>
<thead>
<tr>
<th>LDCPE_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LDCPE_1 is a transparent data latch with data (D), asynchronous clear (CLR), asynchronous preset (PRE), gate enable (GE), and inverted gate (G). When CLR is High, it overrides the other inputs and resets the data (Q) output Low. When PRE is High and CLR is Low, it presets the data (Q) output High. Q reflects the data (D) input while gate enable (GE) is High and gate (G), CLR, and PRE are Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G is High or GE is Low.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-II, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

### Inputs Outputs

<table>
<thead>
<tr>
<th>CLR</th>
<th>PRE</th>
<th>GE</th>
<th>G</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>
Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.

VHDL Inference Code

architecture Behavioral of ldcpe_1 is
begin
process (CLR, D, G, GE, PRE)
begin
  if (CLR='1') then
    Q <= '0';
  elsif (PRE='1') then
    Q <= '1';
  elsif (GE = '1') then
    if (G='0') then
      Q <= D;
    end if;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @(CLR or PRE or D or G or GE) begin
  if (CLR)
    Q <= 0;
  else if (PRE)
    Q <= 1;
  else if (!G && GE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDCPE_1 should be placed
-- after architecture statement but before begin keyword

component LDCPE_1
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    CLR : in STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC;
    GE : in STD_ULOGIC;
    PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDCPE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDCPE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDCPE_1 should be placed
-- in architecture after the begin keyword

LDCPE_1_INSTANCE_NAME : LDCPE_1
-- synthesis translate_off
generic map (  
    INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
    CLR => user_CLR,
    D => user_D,
    G => user_G,
    GE => user_GE,
    PRE => user_PRE);

Verilog Instantiation Template

LDCPE_1 LDCPE_1_instance_name (.Q (user_Q),
    .CLR (user_CLR),
    .D (user_D),
    .G (user_G),
    .GE (user_GE),
    .PRE (user_PRE));

defparam LDCPE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDE

Transparent Data Latch with Gate Enable

Architectures Supported

<table>
<thead>
<tr>
<th>LDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LDE is a transparent data latch with data (D) and gate enable (GE) inputs. Output Q reflects the data (D) while the gate (G) input and gate enable (GE) are High. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains Low.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>G</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of lde is

begin

process (D, G, GE)
begin
    if (GE = '1' and G = '1') then
        Q <= D;
    end if;
end process;

end Behavioral;

Verilog Inference Code

always @(D or G or GE) begin
    if (G && GE)
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDE should be placed
-- after architecture statement but before begin keyword

component LDE
    -- synthesis translate_off
    generic (
        INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
         D : in STD_ULOGIC;
         G : in STD_ULOGIC;
         GE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDE should be placed
-- in architecture after the begin keyword

LDE_INSTANCE_NAME : LDE
    -- synthesis translate_off
    generic map (INIT => bit_value)
    -- synthesis translate_on
    port map (Q => user_Q,
               D => user_D,
               G => user_G,
GE => user_GE);

Verilog Instantiation Template

LDE LDE_instance_name (.Q (user_Q),
    .D (user_D),
    .G (user_G),
    .GE (user_GE));

defparam LDE_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDE_1

Transparent Data Latch with Gate Enable and Inverted Gate

Architectures Supported

<table>
<thead>
<tr>
<th>LDE_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-III</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LDE_1 is a transparent data latch with data (D), gate enable (GE), and inverted gate (G). Output Q reflects the data (D) while the gate (G) input is Low and gate enable (GE) is High. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G is High or GE is Low.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-III, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>G</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of lde_1 is
begin
process (D, G, GE)
begin
  if (GE = '1' and G = '0') then
    Q <= D;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (D or G or GE) begin
  if (!G && GE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDE_1 should be placed
-- after architecture statement but before begin keyword

component LDE_1
  -- synthesis translate_off
  generic (
    INIT : bit := '1');
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC;
    GE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDE_1 should be placed
-- in architecture after the begin keyword

LDE_1_INSTANCE_NAME : LDE_1
  -- synthesis translate_off
  generic map (INIT => bit_value)
  -- synthesis translate_on
  port map (Q => user_Q,
    D => user_D,
    G => user_G,
GE => user_GE);

Verilog Instantiation Template

LDE_1 LDE_1_instance_name (.Q (user_Q),
  .D (user_D),
  .G (user_G),
  .GE (user_GE));

defparam LDE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
**LDG**

Transparent Datagate Latch

**Architectures Supported**

<table>
<thead>
<tr>
<th>LDG</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

LDG is a transparent DataGate latch used for gating input signals to decrease power dissipation. The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The D input(s) of the LDG must be connected to a device input pad(s) and must have no other fan-outs (must not branch). The CPLD fitter maps the G input to the device's DataGate Enable control pin (DGE). There must be no more than one DataGate Enable signal in the design. The DataGate Enable signal may be driven either by a device input pin or any on-chip logic source. The DataGate Enable signal may be reused by other ordinary logic in the design.

The latch is asynchronously cleared, output Low, when power is applied, or when global reset is active. For CPLDs, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. See LDG4, 8, 16 for information on multiple transparent datagate latches for the CoolRunner-II series.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>D</td>
</tr>
</tbody>
</table>

**Usage**

For HDL, this design element can be instantiated.

**VHDL Instantiation Template**

```vhdl
component LDG
```
port (Q : out STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC);
end component;
...
begin
...
INSTANCE_NAME : LDG
    port map (Q => user_Q,
              D => user_D,
              G => user_G);

Verilog Instantiation Template

LDG instance_name (.Q (user_Q),
              .D (user_D),
              .G (user_G));
Multiple Transparent Datagate Latches

Architectures Supported

<table>
<thead>
<tr>
<th>LDG4, LDG8, LDG16</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

LDG4, LDG8, and LDG16 have, respectively, 4, 8, and 16 transparent DataGate latches with a common gate enable (G). These latches are used to gate input signals in order to decrease power dissipation during periods when activity on the input pins is not of interest to the CPLD. The data output (Q) of the latch reflects the data (D) input while the gate enable (G) input is Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The D input(s) of the LDG must be connected to a device input pad(s) and must have no other fan-outs (must not branch). The CPLD fitter maps the G input to the device's DataGate Enable control pin (DGE). There must be no more than one DataGate Enable signal in the design. The DataGate Enable signal may be driven either by a device input pin or any on-chip logic source. The DataGate Enable signal may be reused by other ordinary logic in the design.

The latch is asynchronously cleared, output Low, when power is applied. See LDG for information on single transparent data latches.
LDG8 Implementation
LDP

Transparent Data Latch with Asynchronous Preset

Architectures Supported

<table>
<thead>
<tr>
<th>LDP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

LDP is a transparent data latch with asynchronous preset (PRE). When the PRE input is High, it overrides the other inputs and presets the data (Q) output High. Q reflects the data (D) input while gate (G) input is High and PRE is Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains Low.

The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldp is
begin
    process (PRE, G)
    begin
        if (PRE='1') then
            Q <= "1";
        elsif (G='1') then
            Q <= D;
        end if;
    end process;
end Behavioral;

Verilog Inference Code

always @ (PRE or D or G) begin
    if (PRE)
        Q <= 1;
    else if (G)
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDP should be placed
-- after architecture statement but before begin keyword

component LDP
    -- synthesis translate_off
    generic (    INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
        D : in STD_ULOGIC;
        G : in STD_ULOGIC;
        PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDP
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDP_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDP should be placed
-- in architecture after the begin keyword

LDP_INSTANCE_NAME : LDP
    -- synthesis translate_off
    generic map (    INIT => bit_value)
    -- synthesis translate_on
    port map (Q => user_Q,
        D => user_D,
G => user_G,
PRE => user_PRE);

Verilog Instantiation Template

LDP LDP_instance_name (.Q (user_Q),
  .D (user_D),
  .G (user_G),
  .PRE (user_PRE));

defparam LDP_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDP_1

Transparent Data Latch with Asynchronous Preset and Inverted Gate

Architectures Supported

<table>
<thead>
<tr>
<th>LDP_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LDP_1 is a transparent data latch with asynchronous preset (PRE) and inverted gate (G). When the PRE input is High, it overrides the other inputs and presets the data (Q) output High. Q reflects the data (D) input while gate (G) input and PRE are Low. The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High.

The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldp_1is
begin
process (PRE, G)
begin
if (PRE='1') then
Q <= "1111";
elseif (G='0') then
Q <= D;
end if;
end process;
end Behavioral;

Verilog Inference Code
always @(PRE or D or G) begin
if (PRE)
Q <= 1;
else if (!G)
Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDP_1 should be placed
-- after architecture statement but before begin keyword

component LDP_1
-- synthesis translate_off
generic (    INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
    D : in STD_ULOGIC;
    G : in STD_ULOGIC;
    PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDP_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDP_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDP_1 should be placed
-- in architecture after the begin keyword

LDP_1_INSTANCE_NAME : LDP_1
-- synthesis translate_off
generic map (    INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
    D => user_D,
G => user_G,
PRE => user_PRE);

Verilog Instantiation Template

LDP_1 LDP_1_instance_name (.Q (user_Q),
               .D (user_D),
               .G (user_G),
               .PRE (user_PRE));

defparam LDP_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LDPE

Transparent Data Latch with Asynchronous Preset and Gate Enable

Architectures Supported

<table>
<thead>
<tr>
<th>LDPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LDPE is a transparent data latch with asynchronous preset and gate enable. When the asynchronous preset (PRE) is High, it overrides the other input and presets the data (Q) output High. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output remains unchanged as long as G or GE remains Low.

The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>GE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldpe is
begin

process (D, G, GE, PRE)
begin
  if (PRE='1') then
    Q <= '1';
  elsif (GE = '1') then
    if (G='1') then
      Q <= D;
    end if;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @(PRE or D or G or GE) begin
  if (PRE)
    Q <= 1;
  else if (G && GE)
    Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDPE should be placed
-- after architecture statement but before begin keyword

component LDPE
-- synthesis translate_off
generic (
  INIT : bit := '1');
-- synthesis translate_on
port (Q : out STD_ULOGIC;
  D : in STD_ULOGIC;
  G : in STD_ULOGIC;
  GE : in STD_ULOGIC;
  PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDPE
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDPE_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDPE should be placed
-- in architecture after the begin keyword

LDPE_INSTANCE_NAME : LDPE
-- synthesis translate_off
generic map (
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,
          D => user_D,
          G => user_G,
          GE => user_GE,
          PRE => user_PRE);

Verilog Instantiation Template

LDPE LDPE_instance_name (.Q (user_Q),
                          .D (user_D),
                          .G (user_G),
                          .GE (user_GE),
                          .PRE (user_PRE));

defparam LDPE_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
**LDPE_1**

**Transparent Data Latch with Asynchronous Preset, Gate Enable, and Inverted Gate**

### Architectures Supported

<table>
<thead>
<tr>
<th>LDPE_1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

LDPE_1 is a transparent data latch with asynchronous preset, gate enable, and inverted gate. When the asynchronous preset (PRE) is High, it overrides the other input and presets the data (Q) output High. Q reflects the data (D) input while the gate (G) and PRE are Low and gate enable (GE) is High.

The data on the D input during the Low-to-High gate transition is stored in the latch. The data on the Q output remains unchanged as long as G remains High or GE remains Low.

The latch is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

### Usage

This design element typically should be inferred in the design code; however, the element can be instantiated for cases where strict placement control, relative placement control, or initialization attributes need to be applied.
VHDL Inference Code

architecture Behavioral of ldpe_1 is

begin

process (D, G, GE, PRE)
begin
    if (PRE='1') then
        Q <= '1';
    elseif (GE = '1') then
        if (G='0') then
            Q <= D;
        end if;
    end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (PRE or D or G or GE) begin
    if (PRE)
        Q <= 1;
    else if (!G && GE)
        Q <= D;
end

VHDL Instantiation Template

-- Component Declaration for LDPE_1 should be placed
-- after architecture statement but before begin keyword

component LDPE_1
    -- synthesis translate_off
    generic (INIT : bit := '1');
    -- synthesis translate_on
    port (Q : out STD_ULOGIC;
        D : in STD_ULOGIC;
        G : in STD_ULOGIC;
        GE : in STD_ULOGIC;
        PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for LDPE_1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LDPE_1_instance_name : label is "0";
-- values can be (0 or 1)

-- Component Instantiation for LDPE_1 should be placed
-- in architecture after the begin keyword

LDPE_1_INSTANCE_NAME : LDPE_1
-- synthesis translate_off
generic map (   
  INIT => bit_value)
-- synthesis translate_on
port map (Q => user_Q,   
  D => user_D,   
  G => user_G,   
  GE => user_GE,   
  PRE => user_PRE);

Verilog Instantiation Template

LDPE_1 LDPE_1_instance_name (.Q (user_Q),   
  .D (user_D),   
  .G (user_G),   
  .GE (user_GE),   
  .PRE (user_PRE));

defparam LDPE_1_instance_name.INIT = bit_value;

Commonly Used Constraints

INIT
LUT1, 2, 3, 4

1-, 2-, 3-, 4-Bit Look-Up-Table with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>LUT1, LUT2, LUT3, LUT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LUT1, LUT2, LUT3, and LUT4 are, respectively, 1-, 2-, 3-, and 4-bit look-up-tables (LUTs) with general output (O).

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

LUT1 provides a look-up-table version of a buffer or inverter.

LUTs are the basic Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, Spartan-IIE, and Spartan-3 building blocks. Two LUTs are available in each CLB slice; four LUTs are available in each CLB. The variants, “LUT1_D, LUT2_D, LUT3_D, LUT4_D” and “LUT1_L, LUT2_L, LUT3_L, LUT4_L” provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

LUT3 Function Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = binary equivalent of the hexadecimal number assigned to the INIT attribute
Usage

LUTs are generally inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate LUTs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.

VHDL Instantiation Template for LUT1

```vhdl
-- Component Declaration for LUT1 should be placed
-- after architecture statement but before begin keyword

component LUT1
    -- synthesis translate_off
    generic (    INIT : bit_vector := X"2";    -- synthesis translate_on
    port (O : out STD_ULOGIC;
        IO : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT1
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT1_instance_name : label is "2";
-- values can be 0, 1, 2,or 3

-- Component Instantiation for LUT1 should be placed
-- in architecture after the begin keyword

LUT1_INSTANCE_NAME : LUT1
    -- synthesis translate_off
    generic map (    INIT => hex_value
    -- synthesis translate_on
    port map (O => user_O,
        IO => user_IO);
```

Verilog Instantiation Template For LUT1

```verilog
LUT1 LUT1_instance_name (.O (user_O),
                    .IO (user_IO));

defparam LUT1_instance_name.INIT = hex_value;
```

VHDL Instantiation Template for LUT2

```vhdl
-- Component Declaration for LUT2 should be placed
-- after architecture statement but before begin keyword

component LUT2
    -- synthesis translate_off
    generic (    INIT : bit_vector := X"4";
    -- synthesis translate_on
```
port (O : out STD_ULOGIC;
   I0 : in STD_ULOGIC;
   I1 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT2
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT2_instance_name : label is "4";
-- values can be 0, 1, 2, 3, or 4

-- Component Instantiation for LUT2 should be placed
-- in architecture after the begin keyword

LUT2_INSTANCE_NAME : LUT2
  -- synthesis translate_off
  generic map (
    INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
    I0 => user_I0,
    I1 => user_I1);

Verilog Instantiation Template For LUT2

LUT2 LUT2_instance_name (.O (user_O),
  .I0 (user_I0),
  .I1 (user_I1));

defparam LUT2_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT3

-- Component Declaration for LUT3 should be placed
-- after architecture statement but before begin keyword

component LUT3
  -- synthesis translate_off
  generic (  
    INIT : bit_vector := X"8"));
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
    I0 : in STD_ULOGIC;
    I1 : in STD_ULOGIC;
    I2 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT3
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT3_instance_name : label is "8";
-- values can be 0, 1, 2, 3, 4, 5, 6, 7, 8
-- Component Instantiation for LUT3 should be placed
-- in architecture after the begin keyword --

LUT3_INSTANCE_NAME : LUT3
-- synthesis translate_off
generic map (    
    INIT => hex_value)
-- synthesis translate_on
port map (O => user_O,  
          I0 => user_I0,  
          I1 => user_I1,  
          I2 => user_I2);

Verilog Instantiation Template For LUT3

LUT3 LUT3_instance_name (.O (user_O),  
                        .I0 (user_I0),  
                        .I1 (user_I1),  
                        .I2 (user_I2));
defparam LUT4_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT4

-- Component Declaration for LUT4 should be placed
-- after architecture statement but before begin keyword

component LUT4
-- synthesis translate_off
generic (    
    INIT : bit_vector := X"16");
-- synthesis translate_on
port (O : out STD_ULOGIC;  
      I0 : in STD_ULOGIC;  
      I1 : in STD_ULOGIC;  
      I2 : in STD_ULOGIC;  
      I3 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT4
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT4_instance_name : label is "16";
-- values can be 0 through 16

-- Component Instantiation for LUT4 should be placed
-- in architecture after the begin keyword

LUT4_INSTANCE_NAME : LUT4
-- synthesis translate_off
generic map (    
    INIT => hex_value)
```verilog
-- synthesis translate_on
port map (O => user_0,
    I0 => user_I0,
    I1 => user_I1,
    I2 => user_I2,
    I3 => user_I3);

Verilog Instantiation Template For LUT4

LUT4 LUT4_instance_name (.O (user_0),
    .I0 (user_I0),
    .I1 (user_I1),
    .I2 (user_I2),
    .I3 (user_I3));

defparam LUT4_instance_name.INIT = hex_value;

Commonly Used Constraints

BEL
INIT
LOC
U_SET
```
LUT1_D, LUT2_D, LUT3_D, LUT4_D

1-, 2-, 3-, 4-Bit Look-Up-Table with Dual Output

Architectures Supported

<table>
<thead>
<tr>
<th>LUT1_D, LUT2_D, LUT3_D, LUT4_D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

LUT1_D, LUT2_D, LUT3_D, and LUT4_D are, respectively, 1-, 2-, 3-, and 4-bit look-up-tables (LUTs) with two functionally identical outputs, O and LO. The O output is a general interconnect. The LO output is used to connect to another output within the same CLB slice and to the fast connect buffer.

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

LUT1_D provides a look-up-table version of a buffer or inverter.

See also “LUT1, 2, 3, 4” and “LUT1_L, LUT2_L, LUT3_L, LUT4_L.”

LUT3_D Function Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = binary equivalent of the hexadecimal number assigned to the INIT attribute

Usage

LUTs are generally inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate LUTs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.
VHDL Instantiation Template for LUT1_D

-- Component Declaration for LUT1_D should be placed
-- after architecture statement but before begin keyword

component LUT1_D
    -- synthesis translate_off
    generic (    
        INIT : bit_vector := X"2";    
    -- synthesis translate_on  
    port (LO : out STD_ULOGIC;    
          O : out STD_ULOGIC;    
          I0 : in STD_ULOGIC); 
end component;

-- Component Attribute specification for LUT1_D
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT1_D_instance_name : label is "2";
-- values can be 0, 1, 2, or 3

-- Component Instantiation for LUT1_D should be placed
-- in architecture after the begin keyword

LUT1_D_INSTANCE_NAME : LUT1_D
    -- synthesis translate_off
    generic map (    
        INIT => hex_value)    
    -- synthesis translate_on
    port map (LO => user_LO,    
              O => user_O,    
              I0 => user_IO);

Verilog Instantiation Template For LUT1_D

LUT1_D LUT1_D_instance_name (.LO (user_LO),    
                         .O (user_O),    
                         .IO (user_IO));

defparam LUT1_D_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT2_D

-- Component Declaration for LUT2_D should be placed
-- after architecture statement but before begin keyword

component LUT2_D
    -- synthesis translate_off
    generic (    
        INIT : bit_vector := X"4";    
    -- synthesis translate_on  
    port (LO : out STD_ULOGIC;    
          O : out STD_ULOGIC;    
          IO : in STD_ULOGIC;    
          I1 : in STD_ULOGIC); 
end component;
end component;

-- Component Attribute specification for LUT2_D
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT2_D_instance_name : label is "4";

-- Component Instantiation for LUT2_D should be placed
-- in architecture after the begin keyword

LUT2_D_INSTANCE_NAME : LUT2_D
    -- synthesis translate_off
genetic map (  
        INIT => hex_value)  
    -- synthesis translate_on
    port map (LO => user_LO,  
        I0 => user_I0,  
        I1 => user_I1);

Verilog Instantiation Template For LUT2_D

LUT2_D LUT2_D_instance_name (.LO (user_0),  
    .O (user_0),  
    .I0 (user_I0),  
    .I1 (user_I1));

defparam LUT2_D_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT3_D

-- Component Declaration for LUT3_D should be placed
-- after architecture statement but before begin keyword

component LUT3_D
    -- synthesis translate_off
    generic (  
        INIT : bit_vector := X"8");
    -- synthesis translate_on
    port (LO : out STD_ULOGIC;  
        0 : out STD_ULOGIC;  
        I0 : in STD_ULOGIC;  
        I1 : in STD_ULOGIC;  
        I2 : in STD_ULOGIC);  
end component;

-- Component Attribute specification for LUT3_D
-- should be placed after architecture declaration but
-- before the begin keyword

attribute INIT : string;
attribute INIT of LUT3_D_instance_name : label is "8";

-- Component Instantiation for LUT3_D should be placed
-- in architecture after the begin keyword

LUT3_D_INSTANCE_NAME : LUT3_D
    -- synthesis translate_off
    generic map (
        INIT => hex_value
    )
    -- synthesis translate_on
    port map (LO => user_LO,
               O => user_O,
               I0 => user_I0,
               I1 => user_I1,
               I2 => user_I2);

Verilog Instantiation Template For LUT3_D

LUT3_D LUT3_D_instance_name (.LO (user_LO),
                               .O (user_O),
                               .I0 (user_I0),
                               .I1 (user_I1),
                               .I2 (user_I2));
defparam LUT3_D_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT4_D

-- Component Declaration for LUT4_D should be placed
-- after architecture statement but before begin keyword

component LUT4_D
    -- synthesis translate_off
    generic (    INIT : bit_vector := X"16" );
    -- synthesis translate_on
    port (LO : out STD_ULOGIC;
          O : out STD_ULOGIC;
          I0 : in STD_ULOGIC;
          I1 : in STD_ULOGIC;
          I2 : in STD_ULOGIC;
          I3 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT4_D
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of LUT4_D_instance_name : label is "16";
-- values can be 0 through 16

-- Component Instantiation for LUT4_D should be placed
-- in architecture after the begin keyword

LUT4_D_INSTANCE_NAME : LUT4_D
    -- synthesis translate_off
    generic map (    INIT => hex_value
    )
    -- synthesis translate_on
port map (LO => user_LO,
  O => user_0,
  I0 => user_I0,
  I1 => user_I1,
  I2 => user_I2,
  I3 => user_I3);

Verilog Instantiation Template For LUT4_D

LUT4_D LUT4_D_instance_name (.LO (user_LO),
                           .O (user_0),
                           .I0 (user_I0),
                           .I1 (user_I1),
                           .I2 (user_I2),
                           .I3 (user_I3));

defparam LUT4_D_instance_name.INIT = hex_value;

Commonly Used Constraints

INIT
LOC
RLOC
BEL
U_SET
LUT1_L, LUT2_L, LUT3_L, LUT4_L

1-, 2-, 3-, 4-Bit Look-Up-Table with Local Output

Architectures Supported

<table>
<thead>
<tr>
<th>LUT1_L, LUT2_L, LUT3_L, LUT4_L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

LUT1_L, LUT2_L, LUT3_L, and LUT4_L are, respectively, 1-, 2-, 3-, and 4-bit look-up-tables (LUTs) with a local output (LO) that is used to connect to another output within the same CLB slice and to the fast connect buffer.

A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

LUT1_L provides a look-up-table version of a buffer or inverter.

See also "LUT1, 2, 3, 4" and "LUT1_D, LUT2_D, LUT3_D, LUT4_D."

LUT3_L Function Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

INIT = binary equivalent of the hexadecimal number assigned to the INIT attribute

Usage

LUTs are generally inferred with the logic portions of the HDL code. Xilinx suggests that you instantiate LUTs only if you have a need to implicitly specify the logic mapping, or if you need to manually place or relationally place the logic.
VHDL Instantiation Template for LUT1_L

-- Component Declaration for LUT1_L should be placed after architecture statement but before begin keyword

component LUT1_L
-- synthesis translate_off
generic (
    INIT : bit_vector := X"2";
-- synthesis translate_on
port (LO : out STD_ULOGIC;
     IO : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT1_L
-- should be placed after architecture declaration but before the begin keyword
attribute INIT : string;
attribute INIT of LUT1_L_instance_name : label is "2";
-- values can be 0, 1, 2, or 3

-- Component Instantiation for LUT1_L should be placed in architecture after the begin keyword
LUT1_L_INSTANCE_NAME : LUT1_L
-- synthesis translate_off
generic map (
    INIT => hex_value
-- synthesis translate_on
port map (LO => user_LO,
            IO => user_IO);

Verilog Instantiation Template For LUT1_L

LUT1_L LUT1_L_instance_name (.LO (user_LO),
                                .IO (user_IO));
defparam LUT1_L_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT2_L

-- Component Declaration for LUT2_L should be placed after architecture statement but before begin keyword

component LUT2_L
-- synthesis translate_off
generic (
    INIT : bit_vector := X"4";
-- synthesis translate_on
port (LO : out STD_ULOGIC;
     IO : in STD_ULOGIC;
     I1 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT2_L
-- should be placed after architecture declaration but
attribute INIT : string;
attribute INIT of LUT2_L_instance_name : label is "4";

-- Component Instantiation for LUT2_L should be placed
-- in architecture after the begin keyword
LUT2_L_INSTANCE_NAME : LUT2_L
    -- synthesis translate_off
generic map (    
    INIT => hex_value)
    -- synthesis translate_on
port map (LO => user_LO,
    I0 => user_I0,
    I1 => user_I1);

Verilog Instantiation Template For LUT2_L
LUT2_L LUT2_L_instance_name (.LO (user_LO),
    .I0 (user_I0),
    .I1 (user_I1));
defparam LUT2_L_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT3_L
-- Component Declaration for LUT3_L should be placed
-- after architecture statement but before begin keyword
component LUT3_L
    -- synthesis translate_off
genéric (    
    INIT : bit_vector := X"8");
    -- synthesis translate_on
port (LO : out STD_ULOGIC;
    I0 : in STD_ULOGIC;
    I1 : in STD_ULOGIC;
    I2 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT3_L
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of LUT3_L_instance_name : label is "8";

-- Component Instantiation for LUT3_L should be placed
-- in architecture after the begin keyword --
LUT3_L_INSTANCE_NAME : LUT3_L
    -- synthesis translate_off
genéric map (
INIT => hex_value)
-- synthesis translate_on
port map (LO => user_LO,
       I0 => user_I0,
       I1 => user_I1,
       I2 => user_I2);

Verilog Instantiation Template For LUT3_L

LUT3_L LUT3_L_instance_name (.LO (user_LO),
                      .I0 (user_I0),
                      .I1 (user_I1),
                      .I2 (user_I2));
defparam LUT3_L_instance_name.INIT = hex_value;

VHDL Instantiation Template for LUT4_L

-- Component Declaration for LUT4_L should be placed
-- after architecture statement but before begin keyword --

component LUT4_L
    -- synthesis translate_off
    generic (    INIT : bit_vector := X"16");
    -- synthesis translate_on
    port (LO : out STD_ULOGIC;
          IO : in STD_ULOGIC;
          I1 : in STD_ULOGIC;
          I2 : in STD_ULOGIC;
          I3 : in STD_ULOGIC);
end component;

-- Component Attribute specification for LUT4_L
-- should be placed after architecture declaration but
-- before the begin keyword
attribute INIT : string;
attribute INIT of LUT4_L_instance_name : label is "16";
-- values can be 0 through 16

-- Component Instantiation for LUT4_L should be placed
-- in architecture after the begin keyword --

LUT4_L_INSTANCE_NAME : LUT4_L
-- synthesis translate_off
generic map (    INIT => hex_value)
-- synthesis translate_on
port map (LO => user_LO,
          I0 => user_I0,
          I1 => user_I1,
          I2 => user_I2,
          I3 => user_I3);
Verilog Instantiation Template For LUT4_L

LUT4_L LUT4_L_instance_name (.LO (user_LO),
   .I0 (user_I0),
   .I1 (user_I1),
   .I2 (user_I2),
   .I3 (user_I3));

defparam LUT4_L_instance_name.INIT = hex_value;

Commonly Used Constraints

INIT
LOC
RLOC
BEL
U_SET
M2_1

2-to-1 Multiplexer

Architectures Supported

<table>
<thead>
<tr>
<th>M2_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

The M2_1 multiplexer chooses one data bit from two sources (D1 or D0) under the control of the select input (S0). The output (O) reflects the state of the selected data input. When Low, S0 selects D0 and when High, S0 selects D1.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0, D1, D0</td>
<td>O</td>
</tr>
<tr>
<td>1, 1, X</td>
<td>1</td>
</tr>
<tr>
<td>1, 0, X</td>
<td>0</td>
</tr>
<tr>
<td>0, 0, 1</td>
<td>1</td>
</tr>
<tr>
<td>0, X, 0</td>
<td>0</td>
</tr>
</tbody>
</table>


Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of m2_1 is
begin
process (D0,D1,S0)
begin
  case S0 is
    when '0' => O <= D0;
    when '1' => O <= D1;
    when others => NULL;
  end case;
end process;
end Behavioral;

Verilog Inference Code

always @(D0 or D1 or S0)
begin
  case (S0)
    0 : O <= D0;
    1 : O <= D1;
  endcase
end
M2_1B1

2-to-1 Multiplexer with D0 Inverted

Architectures Supported

<table>
<thead>
<tr>
<th>M2_1B1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

The M2_1B1 multiplexer chooses one data bit from two sources (D1 or D0) under the control of select input (S0). When S0 is Low, the output (O) reflects the inverted value of D0. When S0 is High, O reflects the state of D1.

Inputs Outputs

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>D1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>


Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of m2_1b1 is

begin

process (D0,D1,S0)
begin
    case S0 is
        when '0' => O <= not D0;
        when '1' => O <= D1;
        when others => NULL;
    end case;
end process;
end Behavioral;

Verilog Inference Code

always @(D0 or D1 or S0)
begin
    case (S0)
        0 : O <= !D0;
        1 : O <= D1;
    endcase
end
2-to-1 Multiplexer with D0 and D1 Inverted

Architectures Supported

<table>
<thead>
<tr>
<th>M2_1B2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

The M2_1B2 multiplexer chooses one data bit from two sources (D1 or D0) under the control of select input (S0). When S0 is Low, the output (O) reflects the inverted value of D0. When S0 is High, O reflects the inverted value of D1.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>D1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of m2_1b2 is
begin
process (D0,D1,S0)
begin
  case S0 is
    when '0' => O <= not D0;
    when '1' => O <= not D1;
    when others => NULL;
  end case;
end process;
end Behavioral;

Verilog Inference Code
always @(D0 or D1 or S0)
begin
  case (S0)
    0 : O <= !D0;
    1 : O <= !D1;
  endcase
end
M2_1E

2-to-1 Multiplexer with Enable

Architectures Supported

<table>
<thead>
<tr>
<th>M2_1E</th>
<th>Spartan-II, Spartan-IIe</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

M2_1E is a 2-to-1 multiplexer with enable. When the enable input (E) is High, the M2_1E chooses one data bit from two sources (D1 or D0) under the control of select input (S0). When Low, S0 selects D0 and when High, S0 selects D1. When E is Low, the output is Low.

<table>
<thead>
<tr>
<th>E</th>
<th>S0</th>
<th>D1</th>
<th>D0</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

M2_1E Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II, Spartan-II, Spartan-IIe, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

For HDL, this design element is inferred rather than instantiated.
VHDL Inference Code

architecture Behavioral of m2_1e is
begin
process (D0,D1,E,S0)
begin
if (E='0') then
O <= '0';
else
    case S0 is
    when '0' => O <= D0;
    when '1' => O <= D1;
    when others => NULL;
    end case;
end if;
end process;
end Behavioral;

Verilog Inference Code
begin
if (!E) O <= 0;
else begin
    case (S0)
    0 : O <= D0;
    1 : O <= D1;
    endcase
end
end
M4_1E is a 4-to-1 multiplexer with enable. When the enable input (E) is High, the M4_1E multiplexer chooses one data bit from four sources (D3, D2, D1, or D0) under the control of the select inputs (S1 – S0). The output (O) reflects the state of the selected input as shown in the truth table. When E is Low, the output is Low.

### M4_1E Implementation

XC9500/XC9500XV, XC9500XL

CoolRunner XPLA3

CoolRunner-II

### 4-to-1 Multiplexer with Enable

#### Architectures Supported

<table>
<thead>
<tr>
<th>M4_1E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

#### Truth Table

<table>
<thead>
<tr>
<th>E</th>
<th>S1</th>
<th>S0</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>D0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>D1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>D1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>D2</td>
<td>X</td>
<td>D2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>D3</td>
<td>D3</td>
</tr>
</tbody>
</table>
Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of m4_1e is
begin
  process (D,E,S)
  begin
    if (E='0') then
      O <= '0';
    else
      case S is
      when "00" => O <= D(0);
      when "01" => O <= D(1);
      when "10" => O <= D(2);
      when "11" => O <= D(3);
      when others => NULL;
      end case;
    end if;
  end process;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(D or E or S)
begin
  if (!E)
    O <= 0;
  else
    begin
      case (S)
      0 : O <= D[0];
      1 : O <= D[1];
      2 : O <= D[2];
      3 : O <= D[3];
      endcase
    end
end
```
M8_1E

8-to-1 Multiplexer with Enable

Architectures Supported

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M8_1E</td>
<td></td>
</tr>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

M8_1E is an 8-to-1 multiplexer with enable. When the enable input (E) is High, the M8_1E multiplexer chooses one data bit from eight sources (D7 – D0) under the control of the select inputs (S2 – S0). The output (O) reflects the state of the selected input as shown in the truth table. When E is Low, the output is Low.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>S2</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Dn represents signal on the Dn input; all other data inputs are don’t-cares (X).
M8_1E Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

M8_1E Implementation Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of m8_1e is
begin
process (D, E, S)
begin
  if (E='0') then
    O <= '0';
  else
    case S is
    when "000" => O <= D(0);
    when "001" => O <= D(1);
    when "010" => O <= D(2);
    when "011" => O <= D(3);
    when "100" => O <= D(4);
    when "101" => O <= D(5);
    when "110" => O <= D(6);
    when "111" => O <= D(7);
    when others => NULL;
    end case;
  end if;
end process;
end Behavioral;

Verilog Inference Code
always @(D or E or S)
begin
  if (!E)
    O <= 0;
  else
    begin
      case (S)
        0 : O <= D[0];
        1 : O <= D[1];
        2 : O <= D[2];
        3 : O <= D[3];
        4 : O <= D[4];
        5 : O <= D[5];
        6 : O <= D[6];
        7 : O <= D[7];
      endcase
    end
end
M16_1E

16-to-1 Multiplexer with Enable

Architectures Supported

<table>
<thead>
<tr>
<th>M16_1E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

M16_1E is a 16-to-1 multiplexer with enable. When the enable input (E) is High, the M16_1E multiplexer chooses one data bit from 16 sources (D15 – D0) under the control of the select inputs (S3 – S0). The output (O) reflects the state of the selected input as shown in the truth table. When E is Low, the output is Low.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>E S3 S2 S1 S0 D15 – D0</td>
<td>O</td>
</tr>
<tr>
<td>0 X X X X X</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0 0 D0</td>
<td>D0</td>
</tr>
<tr>
<td>1 0 0 0 1 D1</td>
<td>D1</td>
</tr>
<tr>
<td>1 0 0 1 0 D2</td>
<td>D2</td>
</tr>
<tr>
<td>1 0 1 1 1 D3</td>
<td>D3</td>
</tr>
<tr>
<td>1 1 1 1 0 D12</td>
<td>D12</td>
</tr>
<tr>
<td>1 1 1 0 1 D13</td>
<td>D13</td>
</tr>
<tr>
<td>1 1 1 1 0 D14</td>
<td>D14</td>
</tr>
<tr>
<td>1 1 1 1 1 D15</td>
<td>D15</td>
</tr>
</tbody>
</table>

Dn represents signal on the Dn input; all other data inputs are don't cares (X).

Usage

For HDL, this design element is inferred rather than instantiated.
 VHDL Inference Code

architecture Behavioral of m16_1e is
begin
process (D,E,S)
begin
if (E='0') then
O <= '0';
else
  case S is
  when "0000" => O <= D(0);
  when "0001" => O <= D(1);
  when "0010" => O <= D(2);
  when "0011" => O <= D(3);
  when "0100" => O <= D(4);
  when "0101" => O <= D(5);
  when "0110" => O <= D(6);
  when "0111" => O <= D(7);
  when "1000" => O <= D(8);
  when "1001" => O <= D(9);
  when "1010" => O <= D(10);
  when "1011" => O <= D(11);
  when "1100" => O <= D(12);
  when "1101" => O <= D(13);
  when "1110" => O <= D(14);
  when "1111" => O <= D(15);
  when others => NULL;
  end case;
end if;
end process;
end Behavioral;

 Verilog Inference Code

always @(D or E or S)
begin
if (!E)
    O <= 0;
else
  begin
    case (S)
    0 : O <= D[0];
    1 : O <= D[1];
    2 : O <= D[2];
    3 : O <= D[3];
    4 : O <= D[4];
    5 : O <= D[5];
    6 : O <= D[6];
    7 : O <= D[7];
    8 : O <= D[8];
    9 : O <= D[9];
    10 : O <= D[10];
    11 : O <= D[11];
    12 : O <= D[12];
    13 : O <= D[13];
    end case;
  end
end
14 : O <= D[14];
15 : O <= D[15];
endcase
end
**MULT_AND**

**Fast Multiplier AND**

**Architectures Supported**

<table>
<thead>
<tr>
<th>MULT_AND</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

MULT_AND is an AND component used exclusively for building fast and smaller multipliers. The I1 and I0 inputs must be connected to the I1 and I0 inputs of the associated LUT. The LO output must be connected to the DI input of the associated MUXCY, MUXCY_D, or MUXCY_L.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example Multiplier Using MULT_AND**

**Usage**

For HDL, this design element is instantiated rather than inferred.
VHDL Instantiation Template

-- Component Declaration for MULT_AND should be placed
-- after architecture statement but before begin keyword

component MULT_AND
  port (LO : out STD_ULOGIC;
    I0 : in STD_ULOGIC;
    I1 : in STD_ULOGIC);
end component;

-- Component Attribute specification for MULT_AND
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MULT_AND should be placed
-- in architecture after the begin keyword

MULT_AND_INSTANCE_NAME : MULT_AND
  port map (LO => user_LO,
    I0 => user_I0,
    I1 => user_I1);

Verilog Instantiation Template

MULT_AND MULT_AND_instance_name (.LO (user_LO),
    .I0 (user_I0),
    .I1 (user_I1));

Commonly Used Constraints

U_SET
MULT_STYLE
MULT18X18

18 x 18 Signed Multiplier

Architectures Supported

<table>
<thead>
<tr>
<th>MULT18, MULT18X</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

MULT18X18 is a combinational signed 18-bit by 18-bit multiplier. The value represented in the 18-bit input A is multiplied by the value represented in the 18-bit input B. Output P is the 36-bit product of A and B.

A, B, and P are two's complement.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

XST, Synplify, Exemplar and Synopsys all have the ability to infer the MULT18X18.

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```
-- Component Declaration for MULT18X18 should be placed after architecture statement but before begin keyword
component MULT18X18
    port (P : out STD_LOGIC_VECTOR (35 downto 0);
          A : in STD_LOGIC_VECTOR (17 downto 0);
          B : in STD_LOGIC_VECTOR (17 downto 0));
end component;

-- Component Attribute specification for MULT18X18
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MULT18X18 should be placed
-- in architecture after the begin keyword
MULT18X18_INSTANCE_NAME : MULT18X18
```

port map (P => user_P,
        A => user_A,
        B => user_B);

Verilog Instantiation Template
MULT18X18 MULT18X18_instance_name (.P (user_P),
    .A (user_A),
    .B (user_B));

Commonly Used Constraints
MULT_STYLE
MULT18X18S

18 x 18 Signed Multiplier -- Registered Version

Architectures Supported

<table>
<thead>
<tr>
<th>MULTI18X, MULTI18S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MULT18X18S is the registered version of the 18 x 18 signed multiplier with output P and inputs A, B, C, CE, and R. The registers are initialized to 0 after the GSR pulse.

The value represented in the 18-bit input A is multiplied by the value represented in the 18-bit input B. Output P is the 36-bit product of A and B.

A, B, and P are two's complement.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>CE</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element is instantiated rather than inferred.

VHDL Instantiation Template

-- Component Declaration for MULTI18X18S should be placed
-- after architecture statement but before begin keyword

component MULTI18X18S
    port (P : out STD_LOGIC_VECTOR (35 downto 0); 
            A : in STD_LOGIC_VECTOR (17 downto 0); 
            B : in STD_LOGIC_VECTOR (17 downto 0); 
            C : in STD_ULOGIC; 
            CE : in STD_ULOGIC; 
            R : in STD_ULOGIC); 
end component;

-- Component Attribute specification for MULTI18X18S
-- should be placed after architecture declaration but
-- before the begin keyword
-- Attributes should be placed here

-- Component Instantiation for MULT18X18S should be placed
-- in architecture after the begin keyword

MULT18X18S_INSTANCE_NAME : MULT18X18S
    port map (P => user_P,
               A => user_A,
               B => user_B,
               CE => user_CE,
               C => user_C,
               R => user_R);

Verilog Instantiation Template

MULT18X18S MULT18X18S_instance_name (.P (user_P),
                                      .A (user_A),
                                      .B (user_B),
                                      .C (user_C),
                                      .CE (user_CE),
                                      .R (user_R));

Commonly Used Constraints

HU_SET
MULT_STYLE
MUXCY

2-to-1 Multiplexer for Carry Logic with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXCY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-II</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

MUXCY is used to implement a 1-bit high-speed carry propagate function. One such function can be implemented per logic cell (LC), for a total of:

- 2-bits per CLB for Virtex, Virtex-E, Spartan-II, and Spartan-IIIE
- 4-bits per configurable logic block (CLB) for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X

The direct input (DI) of an LC is connected to the DI input of the MUXCY. The carry in (CI) input of an LC is connected to the CI input of the MUXCY. The select input (S) of the MUXCY is driven by the output of the lookup table (LUT) and configured as a MUX function. The carry out (O) of the MUXCY reflects the state of the selected input and implements the carry out function of each LC. When Low, S selects DI; when High, S selects CI.

The variants, “MUXCY_D” and “MUXCY_L” provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>DI</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.
VHDL Instantiation Template

-- Component Declaration for MUXCY should be placed
-- after architecture statement but before begin keyword

component MUXCY
    port (O : out STD_ULOGIC;
         CI : in STD_ULOGIC;
         DI : in STD_ULOGIC;
         S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXCY
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXCY should be placed
-- in architecture after the begin keyword

MUXCY_INSTANCE_NAME : MUXCY
    port map (O => user_O,
              CI => user_CI,
              DI => user_DI,
              S => user_S);

Verilog Instantiation Template

MUXCY MUXCY_instance_name (.O (user_O),
                           .CI (user_CI),
                           .DI (user_DI),
                           .S (user_S));

Commonly Used Constraints

U_SET
**MUXCY_D**

2-to-1 Multiplexer for Carry Logic with Dual Output

**Architectures Supported**

<table>
<thead>
<tr>
<th>MUXCY_D</th>
<th>Spartan-II, Spartan-IIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

MUXCY_D is used to implement a 1-bit high-speed carry propagate function. One such function can be implemented per logic cell (LC), for a total of 4-bits per configurable logic block (CLB). The direct input (DI) of an LC is connected to the DI input of the MUXCY_D. The carry in (CI) input of an LC is connected to the CI input of the MUXCY_D. The select input (S) of the MUX is driven by the output of the lookup table (LUT) and configured as an XOR function. The carry out (O and LO) of the MUXCY_D reflects the state of the selected input and implements the carry out function of each LC. When Low, S selects DI; when High, S selects CI.

Outputs O and LO are functionally identical. The O output is a general interconnect. See also “MUXCY” and “MUXCY_L”

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>O</td>
</tr>
<tr>
<td>DI</td>
<td>LO</td>
</tr>
<tr>
<td>CI</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

For HDL, this design element can only be instantiated.
VHDL Instantiation Template

-- Component Declaration for MUXCY_D should be placed
-- after architecture statement but before begin keyword

component MUXCY_D
  port (LO : out STD_ULOGIC;
       O  : out STD_ULOGIC;
       CI : in  STD_ULOGIC;
       DI : in  STD_ULOGIC;
       S  : in  STD_ULOGIC);
end component;

-- Component Attribute specification for MUXCY_D
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXCY_D should be placed
-- in architecture after the begin keyword

MUXCY_D_INSTANCE_NAME : MUXCY_D
  port map (LO => user_O,
            O  => user_O,
            CI => user_CI,
            DI => user_DI,
            S  => user_S);

Verilog Instantiation Template

MUXCY_D MUXCY_D_instance_name (.LO (user_LO),
                                .O (user_O),
                                .CI (user_CI),
                                .DI (user_DI),
                                .S (user_S));

Commonly Used Constraints

U_SET
2-to-1 Multiplexer for Carry Logic with Local Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXCY_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXCY_L is used to implement a 1-bit high-speed carry propagate function. One such function can be implemented per logic cell (LC), for a total of 4-bits per configurable logic block (CLB). The direct input (DI) of an LC is connected to the DI input of the MUXCY_L. The carry in (CI) input of an LC is connected to the CI input of the MUXCY_L. The select input (S) of the MUXCY_L is driven by the output of the lookup table (LUT) and configured as an XOR function. The carry out (LO) of the MUXCY_L reflects the state of the selected input and implements the carry out function of each LC. When Low, S selects DI; when High, S selects CI.

See also “MUXCY” and “MUXCY_D”

### Inputs

<table>
<thead>
<tr>
<th>S</th>
<th>DI</th>
<th>CI</th>
<th>LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Usage

For HDL, this design element can only be instantiated.
VHDL Instantiation Template

-- Component Declaration for MUXCY_L should be placed
-- after architecture statement but before begin keyword

component MUXCY_L
  port (LO : out STD_ULOGIC;
       CI : in STD_ULOGIC;
       DI : in STD_ULOGIC;
       S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXCY_L
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXCY_L should be placed
-- in architecture after the begin keyword

MUXCY_L_INSTANCE_NAME : MUXCY_L
  port map (LO => user_O,
            CI => user_CI,
            DI => user_DI,
            S => user_S);

Verilog Instantiation Template

MUXCY_L MUXCY_L_instance_name (.LO (user_LO),
                                 .CI (user_CI),
                                 .DI (user_DI),
                                 .S (user_S));

Commonly Used Constraints

U_SET
MUXF5

2-to-1 Lookup Table Multiplexer with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF5 provides a multiplexer function in a CLB slice for creating a function-of-5 lookup table or a 4-to-1 multiplexer in combination with the associated lookup tables. The local outputs (LO) from the two lookup tables are connected to the I0 and I1 inputs of the MUXF5. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, “MUXF5_D” and “MUXF5_L”, provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF5 should be placed
-- after architecture statement but before begin keyword

component MUXF5
    port (O : out STD_ULOGIC;
         I0 : in STD_ULOGIC;
         I1 : in STD_ULOGIC;
         S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXF5
```
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF5 should be placed
-- in architecture after the begin keyword

MUXF5_INSTANCE_NAME : MUXF5
    port map (O => user_O,
    I0 => user_I0,
    I1 => user_I1,
    S => user_S);

Verilog Instantiation Template

MUXF5 MUXF5_instance_name (.O (user_O),
    .I0 (user_I0),
    .I1 (user_I1),
    .S (user_S));
MUXF5_D

2-to-1 Lookup Table Multiplexer with Dual Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF5_D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF5_D provides a multiplexer function in a CLB slice for creating a function-of-5 lookup table or a 4-to-1 multiplexer in combination with the associated lookup tables. The local outputs (LO) from the two lookup tables are connected to the I0 and I1 inputs of the MUXF5. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output is used to connect to other inputs within the same CLB slice.

See also “MUXF5” and “MUXF5_L”

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF5_D should be placed
-- after architecture statement but before begin keyword

cOMPONENT MUXF5_D
  PORT (LO : OUT STD_ULOGIC;
         O : OUT STD_ULOGIC;
         I0 : IN STD_ULOGIC;
         I1 : IN STD_ULOGIC;
         S  : IN STD_ULOGIC);
END COMPONENT;
```
-- Component Attribute specification for MUXF5_D
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF5_D should be placed
-- in architecture after the begin keyword

MUXF5_D_INSTANCE_NAME : MUXF5_D
    port map (LO => user_LO,
               O  => user_O,
               I0 => user_I0,
               I1 => user_I1,
               S => user_S);

Verilog Instantiation Template

MUXF5_D MUXF5_D_instance_name (.LO (user_LO),
                                  .O  (user_O),
                                  .I0 (user_I0),
                                  .I1 (user_I1),
                                  .S (user_S));
MUXF5_L

2-to-1 Lookup Table Multiplexer with Local Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF5_L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

MUXF5_L provides a multiplexer function in a CLB slice for creating a function-of-5 lookup table or a 4-to-1 multiplexer in combination with the associated lookup tables. The local outputs (LO) from the two lookup tables are connected to the I0 and I1 inputs of the MUXF5. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The LO output is used to connect to other inputs within the same CLB slice.

See also “MUXF5” and “MUXF5_D”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S )</td>
<td>( I0 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>( X )</td>
</tr>
<tr>
<td>1</td>
<td>( X )</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF5_L should be placed
-- after architecture statement but before begin keyword

component MUXF5_L
    port (LO : out STD_ULOGIC;
    I0 : in STD_ULOGIC;
    I1 : in STD_ULOGIC;
    S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXF5_L
```
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF5_L should be placed
-- in architecture after the begin keyword

MUXF5_L_INSTANCE_NAME : MUXF5_L
  port map (LO => user_LO,
            I0 => user_I0,
            I1 => user_I1,
            S => user_S);

Verilog Instantiation Template

MUXF5_L MUXF5_L_instance_name (.LO (user_LO),
                                .I0 (user_I0),
                                .I1 (user_I1),
                                .S (user_S));
MUXF6

2-to-1 Lookup Table Multiplexer with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF6 provides a multiplexer function in a full Virtex, Virtex-E, Spartan-II, or Spartan-IIE CLB, or one half of a Spartan-3, Virtex-II, Virtex-II Pro, or Virtex-II Pro X CLB (two slices) for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I0 and I1 inputs of the MUXF6. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, “MUXF6_D” and “MUXF6_L”, provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```
-- Component Declaration for MUXF6 should be placed
-- after architecture statement but before begin keyword

component MUXF6
    port (O : out STD_ULOGIC;
          I0 : in STD_ULOGIC;
          I1 : in STD_ULOGIC;
          S  : in STD_ULOGIC);
```
end component;

-- Component Attribute specification for MUXF6
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF6 should be placed
-- in architecture after the begin keyword

MUXF6_INSTANCE_NAME : MUXF6
    port map (O => user_O,
               I0 => user_I0,
               I1 => user_I1,
               S => user_S);

Verilog Instantiation Template

MUXF6 MUXF6_instance_name (.O (user_O),
                           .I0 (user_I0),
                           .I1 (user_I1),
                           .S (user_S));
MUXF6_D

2-to-1 Lookup Table Multiplexer with Dual Output

**Architectures Supported**

<table>
<thead>
<tr>
<th>MUXF6_D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF6_D provides a multiplexer function in a full Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, Spartan-IIIE, or Spartan-3 CLB, or one half of a Spartan-3, Virtex-II, Virtex-II Pro, or Virtex-II Pro X CLB (two slices) for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I0 and I1 inputs of the MUXF6. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output is used to connect to other inputs within the same CLB slice.

See also “MUXF6” and “MUXF6_L”

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>O</td>
</tr>
<tr>
<td>I0</td>
<td>LO</td>
</tr>
<tr>
<td>I1</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

For HDL, this design element can only be instantiated.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for MUXF6_D should be placed
-- after architecture statement but before begin keyword

component MUXF6_D
  port (LO : out STD_ULOGIC;
        O  : out STD_ULOGIC;
        I0 : in STD_ULOGIC;
        I1 : in STD_ULOGIC;
```


S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXF6_D
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF6_D should be placed
-- in architecture after the begin keyword

MUXF6_D_INSTANCE_NAME : MUXF6_D
port map (LO => user_LO,
O => user_O,
I0 => user_I0,
I1 => user_I1,
S => user_S);

Verilog Instantiation Template

MUXF6_D MUXF6_D_instance_name (.LO (user_LO),
.O (user_O),
.I0 (user_I0),
.I1 (user_I1),
.S (user_S));
MUXF6_L

2-to-1 Lookup Table Multiplexer with Local Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF6_L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

MUXF6_L provides a multiplexer function in a full Virtex, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, Spartan-II E, or Spartan-3 CLB, or one half of a Spartan-3, Virtex-II, Virtex-II Pro, or Virtex-II Pro X CLB (two slices) for creating a function-of-6 lookup table or an 8-to-1 multiplexer in combination with the associated four lookup tables and two MUXF5s. The local outputs (LO) from the two MUXF5s in the CLB are connected to the I₀ and I₁ inputs of the MUXF6. The S input is driven from any internal net. When Low, S selects I₀. When High, S selects I₁.

The LO output is used to connect to other inputs within the same CLB slice.

See also “MUXF6” and “MUXF6_D”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S I₀ I₁</td>
<td>LO</td>
</tr>
<tr>
<td>0 1 X</td>
<td>1</td>
</tr>
<tr>
<td>0 0 X</td>
<td>0</td>
</tr>
<tr>
<td>1 X 1</td>
<td>1</td>
</tr>
<tr>
<td>1 X 0</td>
<td>0</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF6_L should be placed
-- after architecture statement but before begin keyword

component MUXF6_L
    port (LO : out STD_ULOGIC;
          I₀ : in STD_ULOGIC;
          I₁ : in STD_ULOGIC;
          S : in STD_ULOGIC);
end component;
```
-- Component Attribute specification for MUXF6_L
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF6_L should be placed
-- in architecture after the begin keyword

MUXF6_L_INSTANCE_NAME : MUXF6_L
  port map (LO => user_LO,
             I0 => user_I0,
             I1 => user_I1,
             S => user_S);

Verilog Instantiation Template

MUXF6_L MUXF6_L_instance_name (.LO (user_LO),
                                .I0 (user_I0),
                                .I1 (user_I1),
                                .S (user_S));
MUXF7

2-to-1 Lookup Table Multiplexer with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF7 provides a multiplexer function in a full Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X CLB for creating a function-of-7 lookup table or a 16-to-1 multiplexer in combination with the associated lookup tables. Local outputs (LO) of MUXF6 are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The variants, “MUXF7_D” and “MUXF7_L”, provide additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF7 should be placed
-- after architecture statement but before begin keyword

component MUXF7
    port (O : out STD_ULOGIC;
         I0 : in STD_ULOGIC;
         I1 : in STD_ULOGIC;
         S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXF7
```
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF7 should be placed
-- in architecture after the begin keyword

MUXF7_INSTANCE_NAME : MUXF7
    port map (O => user_O,
               I0 => user_I0,
               I1 => user_I1,
               S => user_S);

Verilog Instantiation Template

MUXF7 MUXF7_instance_name (.O (user_O),
                           .I0 (user_I0),
                           .I1 (user_I1),
                           .S (user_S));
MUXF7_D

2-to-1 Lookup Table Multiplexer with Dual Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF7_D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF7_D provides a multiplexer function in one full Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X CLB for creating a function-of-7 lookup table or a 16-to-1 multiplexer in combination with the associated lookup tables. Local outputs (LO) of MUXF6 are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

Outputs O and LO are functionally identical. The O output is a general interconnect. The LO output is used to connect to other inputs within the same CLB slice.

See also “MUXF7” and “MUXF7_L”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S )</td>
<td>( I0 )</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>( X )</td>
</tr>
<tr>
<td>( X )</td>
<td>0</td>
</tr>
<tr>
<td>( X )</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF7_D should be placed
-- after architecture statement but before begin keyword

component MUXF7_D
    port (LO : out STD_ULOGIC;
         O  : out STD_ULOGIC;
         I0 : in  STD_ULOGIC;
         I1 : in  STD_ULOGIC;
         S  : in  STD_ULOGIC);
end component;
```
-- Component Attribute specification for MUXF7_D
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF7_D should be placed
-- in architecture after the begin keyword

MUXF7_D_INSTANCE_NAME : MUXF7_D
    port map (LO => user_LO,
              O  => user_O,
              I0 => user_I0,
              I1 => user_I1,
              S  => user_S);

Verilog Instantiation Template

MUXF7_D MUXF7_D_instance_name (.LO (user_LO),
                                .O  (user_O),
                                .I0  (user_I0),
                                .I1  (user_I1),
                                .S  (user_S));
MUXF7_L

2-to-1 Lookup Table Multiplexer with Local Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF7_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

MUXF7_L provides a multiplexer function in a full Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X CLB for creating a function-of-7 lookup table or a 16-to-1 multiplexer in combination with the associated lookup tables. Local outputs (LO) of MUXF6 are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The LO output is used to connect to other inputs within the same CLB slice.

See also “MUXF7” and “MUXF7_D”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```vhdl
-- Component Declaration for MUXF7_L should be placed
-- after architecture statement but before begin keyword

component MUXF7_L

port (LO : out STD_ULOGIC;
I0 : in STD_ULOGIC;
I1 : in STD_ULOGIC;
S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXF7_L
```
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for MUXF7_L should be placed
-- in architecture after the begin keyword

MUXF7_L_INSTANCE_NAME : MUXF7_L
  port map (LO => user_LO,
           I0 => user_I0,
           I1 => user_I1,
           S => user_S);

Verilog Instantiation Template

MUXF7_L MUXF7_L_instance_name (.LO (user_LO),
                                .I0 (user_I0),
                                .I1 (user_I1),
                                .S (user_S));
MUXF8

2-to-1 Lookup Table Multiplexer with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>MUXF8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

MUXF8 provides a multiplexer function in two full Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X CLBs for creating a function-of-8 lookup table or a 32-to-1 multiplexer in combination with the associated lookup tables, MUXF5s, MUXF6s, and MUXF7s. Local outputs (LO) of MUXF7 are connected to the I0 and I1 inputs of the MUXF8. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Usage

For HDL, this design element can only be instantiated.

VHDL Instantiation Template

```
-- Component Declaration for MUXF8 should be placed
-- after architecture statement but before begin keyword

component MUXF8
    port (O : out STD_ULOGIC;
          I0 : in STD_ULOGIC;
          I1 : in STD_ULOGIC;
          S : in STD_ULOGIC);
end component;

-- Component Attribute specification for MUXF8
-- should be placed after architecture declaration but
-- before the begin keyword
```
-- Attributes should be placed here

-- Component Instantiation for MUXF8 should be placed
-- in architecture after the begin keyword

MUXF8_INSTANCE_NAME : MUXF8
    port map (O => user_O,
    I0 => user_I0,
    I1 => user_I1,
    S => user_S);

Verilog Instantiation Template

MUXF8 MUXF8_instance_name (.O (user_O),
    .I0 (user_I0),
    .I1 (user_I1),
    .S (user_S));
## NAND2-9

2- to 9-Input NAND Gates with Inverted and Non-Inverted Inputs

### Architectures Supported

<table>
<thead>
<tr>
<th>NAND2, NAND3, NAND4, NAND5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAND2B1, NAND2B2, NAND3B1, NAND3B2, NAND3B3, NAND4B1, NAND4B2, NAND4B3, NAND4B4, NAND5B1, NAND5B2, NAND5B3, NAND5B4, NAND5B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAND6, NAND7, NAND8, NAND9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>
NAND Gate Representations

NAND gates of up to five inputs are available in any combination of inverting and non-inverting inputs. NAND gates of six to nine inputs are available with only non-inverting inputs. To invert inputs, use external inverters. Since each input uses a CLB resource, replace gates with unused inputs with gates having the necessary number of inputs.

See “NAND12, 16” for information on additional NAND functions.
NAND8 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

NAND8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

NAND2 through NAND5 are primitives that can be inferred or instantiated. NAND6 through NAND9 are macros which can be inferred.
VHDL Inference Code

architecture Behavioral of nand6 is
begin
process (I0, I1, I2, I3, I4, I5)
begin
O <= not (I0 and I1 and I2 and I3 and I4 and I5);
end process;
end Behavioral;

Verilog Inference Code

Following is a Verilog example of how to infer a NAND6 element:

always @(I0 or I1 or I2 or I3 or I4 or I5)
begin
O <= !(I0 && I1 && I2 && I3 && I4 && I5);
end

VHDL Instantiation Template for NAND5

Following is the VHDL code for NAND5. To instantiate NAND2, remove I2, I3, and I4. To instantiate NAND3, remove I3 and I4. For NAND4, remove I4. NAND2B1, and NAND2B2 have the same code as NAND2. NAND3B1, 3B2, and 3B3 have the same code as NAND3 and so forth.

-- Component Declaration for NAND5 should be placed
-- after architecture statement but before begin keyword

component NAND5
port (O : out STD_ULOGIC;
I0 : in STD_ULOGIC;
I1 : in STD_ULOGIC;
I2 : in STD_ULOGIC;
I3 : in STD_ULOGIC;
I4 : in STD_ULOGIC);
end component;

-- Component Attribute specification for NAND5
-- should be placed after architecture declaration but
-- before the begin keyword
-- Attributes should be placed here

-- Component Instantiation for NAND5 should be placed
-- in architecture after the begin keyword

NAND5_INSTANCE_NAME : NAND5
port map (O => user_O,
I0 => user_I0,
I1 => user_I1,
I2 => user_I2,
I3 => user_I3,
I4 => user_I4);
Verilog Instantiation Template for NAND5

NAND5 NAND5_instance_name (.O (user_0),
   .I0 (user_I0),
   .I1 (user_I1),
   .I2 (user_I2),
   .I3 (user_I3),
   .I4 (user_I4));

Commonly Used Constraints

None
NAND12, 16

12- and 16-Input NAND Gates with Non-Inverted Inputs

Architectures Supported

<table>
<thead>
<tr>
<th>NAND12, NAND16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

The NAND function is performed in the Configurable Logic Block (CLB) function generators for Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X. The 12- and 16-input NAND functions are available only with non-inverting inputs. To invert some or all inputs, use external inverters.

See NAND2-9 for more information on NAND functions.
NAND12 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
NAND16 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
NAND16 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

For HDL, NAND12 and NAND16 are macros that are inferred. See NAND2-9 for more information about inferring NAND gates.

Commonly Used Constraints

None
### NOR2-9

2- to 9-Input NOR Gates with Inverted and Non-Inverted Inputs

#### Architectures Supported

<table>
<thead>
<tr>
<th>NOR2, NOR3, NOR4, NOR5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOR2B1, NOR2B2, NOR3B1, NOR3B2, NOR3B3, NOR4B1, NOR4B2, NOR4B3, NOR4B4, NOR5B1, NOR5B2, NOR5B3, NOR5B4, NOR5B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOR6, NOR7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOR8, NOR9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>
NOR Gate Representations

NOR gates of up to five inputs are available in any combination of inverting and non-inverting inputs. NOR gates of six to nine inputs are available with only non-inverting inputs. To invert some or all inputs, use external inverters. Since each input uses a CLB resource, replace gates with unused inputs with gates having the necessary number of inputs.

See “NOR12, 16” for information on additional NOR functions.
NOR8 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

NOR8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

NOR2 through NOR5 are primitives that can be inferred or instantiated. NOR6 through NOR9 are macros which can be inferred.
VHDL Inference Code

architecture Behavioral of nor6 is
begin

process (I0, I1, I2, I3, I4, I5)
begin
  O <= not (I0 or I1 or I2 or I3 or I4 or I5);
end process;
end Behavioral;

Verilog Inference Code (NOR6)

always @(I0 or I1 or I2 or I3 or I4 or I5)
begin
  O <= !(I0 || I1 || I2 || I3 || I4 || I5);
end

VHDL Instantiation Template for NOR5

Following is the VHDL code for NOR5. To instantiate NOR2, remove I2, I3, and I4. To instantiate NOR3, remove I3 and I4. For NOR4, remove I4. NOR2B1, and NOR2B2 have the same code as NOR2. NOR3B1, 3B2, and 3B3 have the same code as NOR3 and so forth.

-- Component Declaration for NOR5 should be placed
-- after architecture statement but before begin keyword

component NOR5
  port (O : out STD_ULOGIC;
         I0 : in STD_ULOGIC;
         I1 : in STD_ULOGIC;
         I2 : in STD_ULOGIC;
         I3 : in STD_ULOGIC;
         I4 : in STD_ULOGIC);
end component;

-- Component Attribute specification for NOR5
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for NOR5 should be placed
-- in architecture after the begin keyword

NOR5_INSTANCE_NAME : NOR5
  port map (O => user_O,
            I0 => user_I0,
            I1 => user_I1,
            I2 => user_I2,
            I3 => user_I3,
            I4 => user_I4);
Verilog Instantiation Template for NOR5

NOR5 NOR5_instance_name (.O (user_0),
                 .I0 (user_I0),
                 .I1 (user_I1),
                 .I2 (user_I2),
                 .I3 (user_I3),
                 .I4 (user_I4));

Commonly Used Constraints

None
12- and 16-Input NOR Gates with Non-Inverted Inputs

Architectures Supported

<table>
<thead>
<tr>
<th>NOR12, NOR16</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The 12- and 16-input NOR functions are available only with non-inverting inputs. To invert some or all inputs, use external inverters.

See “NOR2-9” for more information on NOR functions.
NOR16 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
Usage

For HDL, NOR12 and NOR16 are macros that can be inferred. See “NOR2-9” for more information about inferring NOR gates.
**OBUF, 4, 8, 16**

**Single- and Multiple-Output Buffers**

**Architectures Supported**

<table>
<thead>
<tr>
<th>OBUF</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBUF, OBUF4, OBUF8, OBUF16</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

OBUF, OBUF4, OBUF8, and OBUF16 are single and multiple output buffers. An OBUF isolates the internal circuit and provides drive current for signals leaving a chip. OBUFs exist in input/output blocks (IOB). The output (O) of an OBUF is connected to an OPAD or an IOPAD.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, if a high impedance (Z) signal from an on-chip 3-state buffer (like BUFE) is applied to the input of an OBUF, it is propagated to the CPLD device output pin.

For Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, see “OBUF_selectIO” for information on OBUF variants with selectable I/O interfaces. The I/O interface standard used by OBUF, 4, 8, and 16 is LVTTL. Also, Virtex, Virtex-E, Spartan-II, and Spartan-IIIE OBUF, 4, 8, and 16 have selectable drive and slew rates using the DRIVE and SLOW or FAST constraints. The defaults are DRIVE=12 mA and SLOW slew.

Usage

OBUFs are typically inferred for all top level input ports, but they can also be instantiated if necessary.

VHDL Instantiation Template

```vhdl
-- Component Declaration for OBUF should be placed
-- after architecture statement but before begin keyword

component OBUF
    port (O : out STD_ULOGIC;
         I : in STD_ULOGIC);
end component;

-- Component Attribute specification for OBUF
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OBUF should be placed
```

OBUF8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
-- in architecture after the begin keyword

OBUF_INSTANCE_NAME : OBUF
    port map (O => user_O,
            I => user_I);

Verilog Instantiation Template

OBUF_instance_name (.O (user_O),
                    .I (user_I));

Commonly Used Constraints

DRIVE
IOSTANDARD
IOBDELAY
SLEW
OBUF_selectIO

Single Output Buffer with Selectable I/O Interface

<table>
<thead>
<tr>
<th>OBUF_selectIO</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-II E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500V, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

For Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, and Spartan-II E, OBUF and its selectIO variants (listed in the "Components" column in the table below) are single output buffers whose I/O interface corresponds to a specific I/O standard. The name extensions (LVCMOS2, PCI33_3, PCI33_5, etc.) specify the standard. The S, F, and 2, 4, 6, 8, 12, 16, 24 extensions specify the slew rate (SLOW or FAST) and the drive power (2, 4, 6, 8, 12, 16, 24 mA) for the LVTTL standard variants. For example, OBUF_F_12 is a single output buffer that uses the LVTTL I/O-signaling standard with a FAST slew and 12mA of drive power. You can attach an IOSTANDARD attribute to an OBUF instance instead of using an OBUF selectIO component. Check marks (✓) in the "Spartan-II, Virtex" and "Spartan-II E, Virtex-E" columns indicate the components and IOSTANDARD attribute values available for those architectures.

For Virtex-II, Virtex-II Pro, and Virtex-II Pro X, an OBUF that uses the LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 signaling standards has selectable drive and slew rates using the DRIVE and SLOW or FAST constraints. The defaults are DRIVE=12 mA and SLOW slew.

For Virtex, Virtex-E, Spartan-II, and Spartan-II E, an OBUF that uses the LVTTL signaling standard has selectable drive and slew rates using the DRIVE and SLOW or FAST constraints. The defaults are DRIVE=12 mA and SLOW slew.

An OBUF isolates the internal circuit and provides drive current for signals leaving a chip. OBUFS exist in input/output blocks (IOB). The output (O) of an OBUF is connected to an OPAD or an IOPAD.

The hardware implementation of the I/O standard requires that you follow a set of usage rules for the SelectIO buffer components. See the “SelectIO Usage Rules” under the IBUF_selectIO section for information on using these components.
## Spartan-II, Spartan-IIIE, Virtex, and Virtex-E OBUF _selectIO_ Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-IIIE, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUF</td>
<td>√</td>
<td>√</td>
<td>(default is LVTTL)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_2</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_4</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_6</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_8</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_12</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_16</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_S_24</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_2</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_4</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_6</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_8</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_12</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_16</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_F_24</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_AGTP</td>
<td>√</td>
<td>√</td>
<td>AGP</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_CTT</td>
<td>√</td>
<td>√</td>
<td>CTT</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_GTL</td>
<td>√</td>
<td>√</td>
<td>GTL</td>
<td>No</td>
</tr>
<tr>
<td>OBUF_GTLP</td>
<td>√</td>
<td>√</td>
<td>GTLP</td>
<td>No</td>
</tr>
<tr>
<td>OBUF_HSTL_I</td>
<td>√</td>
<td>√</td>
<td>HSTL_I</td>
<td>1.5</td>
</tr>
<tr>
<td>OBUF_HSTL_III</td>
<td>√</td>
<td>√</td>
<td>HSTL_III</td>
<td>1.5</td>
</tr>
<tr>
<td>OBUF_HSTL_IV</td>
<td>√</td>
<td>√</td>
<td>HSTL_IV</td>
<td>1.5</td>
</tr>
<tr>
<td>OBUF_LVC莫斯2</td>
<td>√</td>
<td>√</td>
<td>LVC莫斯2</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUF_LVC莫斯18</td>
<td>√</td>
<td>√</td>
<td>LVC莫斯18</td>
<td>1.8</td>
</tr>
<tr>
<td>OBUF_LVDS</td>
<td>√</td>
<td>√</td>
<td>LVDS</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUF_LVPECL</td>
<td>√</td>
<td>√</td>
<td>LVPECL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_PCI33_3</td>
<td>√</td>
<td>√</td>
<td>PCI33_3</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_PCI33_5</td>
<td>√</td>
<td>√^α</td>
<td>PCI33_5</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_PCI66_3</td>
<td>√</td>
<td>√</td>
<td>PCI66_3</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_PCIx66_3</td>
<td>√</td>
<td>√</td>
<td>PCIx66_3</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_SSTL2_1</td>
<td>√</td>
<td>√</td>
<td>SSTL2_1</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUF_SSTL2_II</td>
<td>√</td>
<td>√</td>
<td>SSTL2_II</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUF_SSTL3_1</td>
<td>√</td>
<td>√</td>
<td>SSTL3_1</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUF_SSTL3_II</td>
<td>√</td>
<td>√</td>
<td>SSTL3_II</td>
<td>3.3</td>
</tr>
</tbody>
</table>

^α Not supported for Virtex-E.

^b The LVTTL attribute also requires a slew value (FAST or SLOW) and DRIVE value. See the FAST, SLOW, and DRIVE attribute descriptions in the Xilinx Constraints Guide for valid values for each architecture.
The Virtex-II, Virtex-II Pro, and Virtex-II Pro X library includes some OBUF_selectIO components for compatibility with older, existing designs and other architectures. For new Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs, however, the recommended method for using OBUF SelectIO buffers is to attach an IOSTANDARD attribute to an OBUF component. For example, attach IOSTANDARD=GTL to an OBUF instead of using the OBUF_GTLP component for new Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs. The IOSTANDARD attributes that can be attached to an OBUF component are listed in the “IOSTANDARD (Attribute Value)” column.

Attach an IOSTANDARD attribute to an OBUF and assign the value indicated in the “IOSTANDARD (Attribute Value)” column to program the output for the I/O standard associated with that value.

The LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33 attributes also require a slew value (FAST or SLOW) and DRIVE value. See the FAST, SLOW, and DRIVE attribute descriptions in the Xilinx Constraints Guide.

### Virtex-II, Virtex-II Pro, Virtex-II Pro X OBUF_selectIO IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Drive</th>
<th>Slew</th>
<th>Termination Type Output</th>
<th>VREF Input *</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGP</td>
<td>√</td>
<td>√</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.32</td>
<td>3.3</td>
</tr>
<tr>
<td>GTL</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.8</td>
<td>No</td>
</tr>
<tr>
<td>GTL_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>.8</td>
<td>1.2</td>
</tr>
<tr>
<td>GTLP</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.0</td>
<td>No</td>
</tr>
<tr>
<td>GTLP_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.90</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_II</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_II_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_II_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>Split</td>
<td>.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_II_DCI_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>Split</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_III</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_III_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_III_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_III_DCI_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_IV</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_IV_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_IV_DCI</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_IV_DCI_18</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>LVCMOS12^a</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>2, 4, 6, 8</td>
<td>Fast/Slow</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>LVCMOS15^a</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>2, 4, 6, 8, 12, 16</td>
<td>Fast/Slow</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>LVCMOS18^a</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>2, 4, 6, 8, 12, 16</td>
<td>Fast/Slow</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>LVCMOS25^a</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>2, 4, 6, 8, 12, 16, 24</td>
<td>Fast/Slow</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>LVCMOS33^a</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>2, 4, 6, 8, 12, 16, 24</td>
<td>Fast/Slow</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>LVDCI_15</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LVDCI_18</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Virtex-II, Virtex-II Pro, Virtex-II Pro X OBUF_selectIO IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Drive</th>
<th>Slew</th>
<th>Termination Type Output</th>
<th>VREF Input</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVDCI_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDCI_33</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>LVDCI_DV2_15</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>1.5</td>
</tr>
<tr>
<td>LVDCI_DV2_18</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>1.8</td>
</tr>
<tr>
<td>LVDCI_DV2_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDCI_DV2_33</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>LVTTLa</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2, 4, 6, 8, 12, 16, 24 Fast/Slow</td>
<td>None</td>
<td>No</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>PCI66_3</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>PCI66_3</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>SSTL18_I</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL18_I_DCII</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL18_II_DCII</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Split</td>
<td>1.25</td>
<td>2.5</td>
</tr>
<tr>
<td>SSTL2_I_DCII</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.25</td>
<td>2.5</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.25</td>
<td>2.5</td>
</tr>
<tr>
<td>SSTL2_II_DCII</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Split</td>
<td>1.25</td>
<td>2.5</td>
</tr>
<tr>
<td>SSTL3_I</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.5</td>
<td>3.3</td>
</tr>
<tr>
<td>SSTL3_I_DCII</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.5</td>
<td>3.3</td>
</tr>
<tr>
<td>SSTL3_II</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.5</td>
<td>3.3</td>
</tr>
<tr>
<td>SSTL3_II_DCII</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Split</td>
<td>1.5</td>
<td>3.3</td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.

Usage

The recommended usage for OBUF_selectIO is to allow the OBUFs be inferred and apply the IOSTANDARD constraint to the input in either the UCF or in the HDL code. OBUF_selectIO can also be instantiated if necessary.

VHDL Instantiation Template

```
-- Component Declaration for OBUF_selectIO should be placed
-- after architecture statement but before begin keyword

component OBUF_selectIO
    port (O : out STD_ULOGIC;
        I : in STD_ULOGIC);
end component;

-- Component Attribute specification for OBUF_selectIO
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OBUF_selectIO should be placed
```
-- in architecture after the begin keyword

OBUF_selectIO_INSTANCE_NAME : OBUF_selectIO
  port map (O => user_O,
          I => user_I);

Verilog Instantiation Template

OBUF_selectIO instance_name (.O (user_O),
                           .I (user_I));

Commonly Used Constraints

DRIVE
IOBDELAY
SLEW
IOSTANDARD
OBUFDS

Differential Signaling Output Buffer with Selectable I/O Interface

Architectures Supported

<table>
<thead>
<tr>
<th>OBUFDS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OBUFDS is a single output buffer that supports low-voltage, differential signaling (1.8v CMOS). OBUFDS isolates the internal circuit and provides drive current for signals leaving the chip. Its output is represented as two distinct ports (O and OB), one deemed the “master” and the other the “slave.” The master and the slave are opposite phases of the same logical signal (for example, MYNET and MYNETB).

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>O</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The IOSTANDARD attribute values listed in the following table can be applied to an OBUFDS component to provide SelectIO interface capability. See the Xilinx Constraints Guide for information using these attributes.

Attach an IOSTANDARD attribute to an OBUFDS and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the output for the I/O standard associated with that value.

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Output</th>
<th>VREF Input</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLVDS_25</td>
<td>√</td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LDT_25</td>
<td>√</td>
<td>√</td>
<td></td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDS_25 (default)</td>
<td>√</td>
<td>√</td>
<td></td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDS_33</td>
<td>√</td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>LVDSEXT_25</td>
<td>√</td>
<td>√</td>
<td></td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDSEXT_33</td>
<td>√</td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>LVPECL_25</td>
<td>√</td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVPECL_33</td>
<td>√</td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>ULVDS_25</td>
<td>√</td>
<td>√</td>
<td></td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
</tbody>
</table>
**Usage**

For HDL, this design element should be instantiated rather than inferred.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for OBUFDS should be placed
-- after architecture statement but before begin keyword

component OBUFDS
  port (O : out STD_ULOGIC;
        OB : out STD_ULOGIC;
        I : in STD_ULOGIC);
end component;

-- Component Attribute specification for OBUFDS
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OBUFDS should be placed
-- in architecture after the begin keyword

OBUFDS_INSTANCE_NAME : OBUFDS
  port map (O => user_O,
            OB => user_OB,
            I => user_I);
```

**Verilog Instantiation Template**

```verilog
OBUFDS instance_name (.O (user_O),
                       .OB (user_OB),
                       .I (user_I));
```

**Commonly Used Constraints**

ISTANDARD, IOBDELAY
OBUFE, 4, 8, 16

3-State Output Buffers with Active-High Output Enable

Architectures Supported

<table>
<thead>
<tr>
<th>OBUFE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Primitive</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBUFE4, OBUFE8, OBUFE16</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

OBUFE, OBUFE4, OBUFE8, and OBUFE16 are 3-state buffers with inputs I, I3 – I0, I7 – I0, and I15-I0, respectively; outputs O, O3 – O0, O7 – O0, and O15-O0, respectively; and active-High output enable (E). When E is High, data on the inputs of the buffers is transferred to the corresponding outputs. When E is Low, the output is High impedance (off or Z state). An OBUFE isolates the internal circuit and provides drive current for signals leaving a chip. An OBUFE output is connected to an OPAD or an IOPAD. An OBUFE input is connected to the internal circuit.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>I</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

OBUFE Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
**OBUFE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X**

**OBUFE8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II, Spartan-II, Spartan-IIe, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X**

**Usage**

For HDL, these design elements are instantiated rather than inferred.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for OBUFE should be placed
-- after architecture statement but before begin keyword

component OBUFE
    port (O : out STD_ULOGIC;
         E : in STD_ULOGIC;
         I : in STD_ULOGIC);
end component;

-- Component Attribute specification for OBUFE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OBUFE should be placed
-- in architecture after the begin keyword

OBUFE_INSTANCE_NAME : OBUFE
```

---

**Diagram**

[Diagram of OBUFE implementation and usage]
port map (O => user_O,
    E => user_E,
    I => user_I);

Verilog Instantiation Template

OBUFE_instance_name (.O (user_O),
    .E (user_E),
    .I (user_I));

Commonly Used Constraints

IOBDELAY
Single and Multiple 3-State Output Buffers with Active-Low Output Enable

Architectures Supported

<table>
<thead>
<tr>
<th>OBUFT</th>
<th>OBUFT4, OBUFT8, OBUFT16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OBUFT, OBUFT4, OBUFT8, and OBUFT16 are single and multiple 3-state output buffers with inputs I, I3 – I0, I7 – I0, I15 – I0, outputs O, O3 – O0, O7 – O0, O15 – O0, and active-Low output enables (T). When T is Low, data on the inputs of the buffers is transferred to the corresponding outputs. When T is High, the output is high impedance (off or Z state). OBUFTs isolate the internal circuit and provide extra drive current for signals leaving a chip. An OBUFT output is connected to an OPAD or an IOPAD.

For Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, Spartan-IIIE, and Spartan-3, see “OBUFT_selectIO” for information on OBUFT variants with selectable I/O interfaces. OBUFT, 4, 8, and 16 use the LVTTL standard. Also, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, Spartan-IIIE, and Spartan-3 OBUFT, 4, 8, and 16 have selectable drive and slew rates using the DRIVE and SLOW or FAST constraints. The defaults are DRIVE=12 mA and SLOW slew.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>OBUFT</th>
<th>OBUFT4</th>
<th>OBUFT16</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>I[15:0]</td>
<td>O[15:0]</td>
<td>X9449</td>
</tr>
<tr>
<td>I</td>
<td>O</td>
<td>O</td>
<td>X9450</td>
</tr>
<tr>
<td>O</td>
<td>I</td>
<td>O</td>
<td>X3817</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

OBUFT8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

For HDL, OBUFT is instantiated rather than inferred.
VHDL Instantiation Template

-- Component Declaration for OBUFT should be placed
-- after architecture statement but before begin keyword

component OBUFT
    port (O : out STD_ULOGIC;
         I : in STD_ULOGIC;
         T : in STD_ULOGIC);
end component;

-- Component Attribute specification for OBUFT
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OBUFT should be placed
-- in architecture after the begin keyword

OBUFT_INSTANCE_NAME : OBUFT
    port map (O => user_O,
              I => user_I,
              T => user_T);

Verilog Instantiation Template

OBUFT_instance_name (.O (user_O),
                     .I (user_I),
                     .T (user_T));

Commonly Used Constraints

DRIVE
IOSTANDARD
IOBDELAY
SLEW
**OBUFT_selectIO**

Single 3-State Output Buffer with Active-Low Output Enable and Selectable I/O Interface

**Architectures Supported**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>OBUFT_selectIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

For Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X, Spartan-II, Spartan-IIIE, and Spartan-3, OBUFT and its selectIO variants (listed in the "Component" column in the table below) are single 3-state output buffers with active-Low output Enable whose I/O interface corresponds to a specific I/O standard. The name extensions (LVCMOS2, PCI33_3, PCI33_5, etc.) specify the standard. The S, F, and 2, 4, 6, 8, 12, 16, 24 extensions specify the slew rate (SLOW or FAST) and the drive power (2, 4, 6, 8, 12, 16, 24 mA) for the LVTTL standard. For example, OBUFT_S_4 is a 3-state output buffer with active-Low output enable that uses the LVTTL I/O signaling standard with a SLOW slew and 4mA of drive power. You can attach an IOSTANDARD attribute to an OBUFT instance instead of using an OBUFT_selectIO component. Check marks (√) in the "Spartan-II, Virtex," and "Spartan-IIIE, Virtex-E" columns indicate the components and IOSTANDARD attribute values available for each architecture.

The hardware implementation of the I/O standards requires that you follow a set of usage rules for the SelectIO buffers. See “SelectIO Usage Rules” under the IBUF_selectIO section for information on using these components and IOSTANDARD attributes.

**Spartan-II, Spartan-IIIE, Virtex, and Virtex-E OBUFT_selectIO Components and IOSTANDARD Attributes**

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-IIIE, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUFT</td>
<td>√</td>
<td>√</td>
<td>defaults to LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_2</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_4</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_6</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_8</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_12</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_16</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_S_24</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_F_2</td>
<td>√</td>
<td>√</td>
<td>LVTTL</td>
<td>3.3</td>
</tr>
</tbody>
</table>
Spartan-II, Spartan-IIIE, Virtex, and Virtex-E OBUFT _selectIO Components and IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>Component</th>
<th>Spartan-II, Virtex</th>
<th>Spartan-IIIE, Virtex-E</th>
<th>IOSTANDARD (Attribute Value)</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUFT_F_4</td>
<td>√</td>
<td>√</td>
<td>LVTTL (^b)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_F_6</td>
<td>√</td>
<td>√</td>
<td>LVTTL (^b)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_F_8</td>
<td>√</td>
<td>√</td>
<td>LVTTL (^b)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_F_12</td>
<td>√</td>
<td>√</td>
<td>LVTTL (^b)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_F_16</td>
<td>√</td>
<td>√</td>
<td>LVTTL (^b)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_F_24</td>
<td>√</td>
<td>√</td>
<td>LVTTL (^b)</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_AGIP</td>
<td>√</td>
<td>√</td>
<td>AGP</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_CTT</td>
<td>√</td>
<td>√</td>
<td>CTT</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_GTL</td>
<td>√</td>
<td>√</td>
<td>GTL</td>
<td>No</td>
</tr>
<tr>
<td>OBUFT_GTPL</td>
<td>√</td>
<td>√</td>
<td>GTLP</td>
<td>No</td>
</tr>
<tr>
<td>OBUFT_HSTL_I</td>
<td>√</td>
<td>√</td>
<td>HSTL_1</td>
<td>1.5</td>
</tr>
<tr>
<td>OBUFT_HSTL_II</td>
<td>√</td>
<td>√</td>
<td>HSTL_II</td>
<td>1.5</td>
</tr>
<tr>
<td>OBUFT_HSTL_III</td>
<td>√</td>
<td>√</td>
<td>HSTL_III</td>
<td>1.5</td>
</tr>
<tr>
<td>OBUFT_LVCMOS2</td>
<td>√</td>
<td>√</td>
<td>LVCMOS2</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUFT_LVCMOS18</td>
<td>√</td>
<td>√</td>
<td>LVCMOS18 (^b)</td>
<td>1.8</td>
</tr>
<tr>
<td>OBUFT_LVDS</td>
<td>√</td>
<td>√</td>
<td>LVDS</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUFT_LVPECL</td>
<td>√</td>
<td>√</td>
<td>LVPECL</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_PCI33_3</td>
<td>√</td>
<td>√</td>
<td>PCI33_3</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_PCI33_5</td>
<td>√</td>
<td>√(^a)</td>
<td>PCI33_5</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_PCI66_3</td>
<td>√</td>
<td>√</td>
<td>PCI66_3</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_PCIE66_3</td>
<td>√</td>
<td>√</td>
<td>PCIE66_3</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_SSTL2_1</td>
<td>√</td>
<td>√</td>
<td>SSTL2_1</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUFT_SSTL2_II</td>
<td>√</td>
<td>√</td>
<td>SSTL2_II</td>
<td>2.5</td>
</tr>
<tr>
<td>OBUFT_SSTL3_1</td>
<td>√</td>
<td>√</td>
<td>SSTL3_1</td>
<td>3.3</td>
</tr>
<tr>
<td>OBUFT_SSTL3_II</td>
<td>√</td>
<td>√</td>
<td>SSTL3_II</td>
<td>3.3</td>
</tr>
</tbody>
</table>

\(^a\) Not supported for Virtex-E.
\(^b\) The LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33 attributes also require a slew value (FAST or SLOW) and DRIVE value. See the FAST, SLOW, and DRIVE attribute descriptions in the Xilinx Constraints Guide for valid values for each architecture.

The Virtex-II, Virtex-II Pro, and Virtex-II Pro X library includes some OBUFT _selectIO components for compatibility with older, existing designs and other architectures. For new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs, however, the recommended method for using OBUFT SelectIO buffers is to attach an IOSTANDARD attribute to an OBUFT component. For example, attach IOSTANDARD=GTLP to an OBUFT instead of using the OBUFT_GTLP component for new Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X designs. The IOSTANDARD attributes that can be attached to an OBUFT component are listed in the "IOSTANDARD (Attribute Value)" column in the following table. See “SelectIO Usage Rules” for information on using these IOSTANDARD attributes.
Attach an IOSTANDARD attribute to an OBUFT and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the output for the I/O standard associated with that value.

The LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33 attributes also require a slew value (FAST or SLOW) and DRIVE value. See the FAST, SLOW, and DRIVE attribute descriptions in the Xilinx Constraints Guide.

### Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X OBUFT_\_selectIO IOSTANDARD Attributes

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Drive</th>
<th>Slew</th>
<th>Termination Type Output</th>
<th>VREF</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGP</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.32</td>
<td>3.3</td>
</tr>
<tr>
<td>GTL</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.8</td>
<td>No</td>
</tr>
<tr>
<td>GTL_DCI</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>GTP</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.0</td>
<td>No</td>
</tr>
<tr>
<td>GTPP_DCI</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.90</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.75</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_Il</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Split</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_Il</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Single</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL_I_DCI</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>.9</td>
<td>1.5</td>
</tr>
<tr>
<td>LVCMS12a</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>LVCMS15a</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>LVCMS18a</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>LVCMS25a</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>LVCMS33a</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>LVDCI_15</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_18</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_25</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_33</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_15</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_18</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_25</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVDCI_DV2_33</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>LVTTLa</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>PCI33_3</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>PCI66_3</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>PCIX</td>
<td>√</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Driver</td>
<td>2.4</td>
<td>3.3</td>
</tr>
</tbody>
</table>
OBUFT and its variants have selectable drive and slew rates using the DRIVE and FAST or SLOW constraints. The defaults are DRIVE=12 mA and SLOW slew.

When T is Low, data on the input of the buffer is transferred to the output. When T is High, the output is high impedance (off or Z state). OBUFTs isolate the internal circuit and provide extra drive current for signals leaving a chip. An OBUFT_selectIO output is connected to an OPAD or an IOPAD.

**Usage**

For HDL, these design elements are instantiated rather than inferred.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for OBUFT_selectIO should be placed
-- after architecture statement but before begin keyword

component OBUFT_selectIO

    port (O : out STD_ULOGIC;
         I : in STD_ULOGIC;
         T : in STD_ULOGIC);

end component;

-- Component Attribute specification for OBUFT_selectIO
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
-- Component Instantiation for OBUFT_selectIO should be
-- placed in architecture after the begin keyword

OBUFT_selectIO_INSTANCE_NAME : OBUFT_selectIO

    port map (O => user_O,
              I => user_I,
```

**Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X OBUFT_selectIO IOSTANDARD Attributes**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Architectures</th>
<th>Attribute Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTANDARD</td>
<td>Spartan-3</td>
<td>Virtex-II</td>
</tr>
<tr>
<td>SSTL18_I</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL18_I_DCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL18_II_DCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL2_I_DCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL2_II_DCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL3_I</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL3_I_DCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL3_II</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SSTL3_II_DCI</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.
Verilog Instantiation Template

OBUFT_selectIO instance_name (.O (user_O),
   .I (user_I),
   .T (user_T));

Commonly Used Constraints

DRIVE, IOBDELAY, SLEW, and IOSTANDARD
OBUFTDS

3-State Differential Signaling Output Buffer with Active Low Output Enable and Selectable I/O Interface

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>OBUFTDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OBUFTDS is a single 3-state, differential signaling output buffer with active Low enable and a Select I/O interface.

When T is Low, data on the input of the buffer is transferred to the output (O) and inverted output (OB). When T is High, both outputs are high impedance (off or Z state).

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>T</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The IOSTANDARD attribute values listed in the following table can be applied to an IBUFGDS component to provide selectIO interface capability. See the Xilinx Constraints Guide for information using these attributes. The hardware implementation of the I/O standards requires that you follow a set of usage rules for the SelectIO buffer components. See the “SelectIO Usage Rules” under the IBUF_selectIO section for information on using these IOSTANDARD attributes.
Attach an IOSTANDARD attribute to an OBUFTDS and assign the value indicated in the "IOSTANDARD (Attribute Value)" column to program the outputs for the I/O standard associated with that value.

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Spartan-3</th>
<th>Virtex-II</th>
<th>Virtex-II Pro, Virtex-II Pro X</th>
<th>Termination Type Output</th>
<th>VREF Input *</th>
<th>Output VCCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLVDS_25</td>
<td></td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LDT_25</td>
<td>√</td>
<td></td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDS_25 (default)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDS_33</td>
<td></td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>LVPECL_25</td>
<td></td>
<td></td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVPECL_33</td>
<td></td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>LVDSEXT_25</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
<tr>
<td>LVDSEXT_33</td>
<td></td>
<td></td>
<td></td>
<td>None</td>
<td>No</td>
<td>3.3</td>
</tr>
<tr>
<td>ULVDS_25</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>None</td>
<td>No</td>
<td>2.5</td>
</tr>
</tbody>
</table>

* VREF requirement when this IOSTANDARD is an input.

Usage

For HDL, these design elements are instantiated rather than inferred.

**VHDL Instantiation Template**

```
-- Component Declaration for OBUFTDS should be placed
-- after architecture statement but before begin keyword

component OBUFTDS
    port (O : out STD_ULOGIC;
        OB : out STD_ULOGIC;
        I : in  STD_ULOGIC;
        T : in  STD_ULOGIC);
end component;

-- Component Attribute specification for OBUFTDS
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OBUFTDS should be
-- placed in architecture after the begin keyword

OBUFTDS_INSTANCE_NAME : OBUFTDS
    port map (O => user_O,
        OB => user_OB,
        I => user_I,
        T => user_T);
```

**Verilog Instantiation Template**

```
OBUFTDS instance_name (.O (user_O),
```


.OB (user_OB),
.I (user_I),
.T (user_T));

Commonly Used Constraints
IOSTANDARD, IOBDELAY
OFD, OFD4, OFD8, OFD16

Single- and Multiple-Output D Flip-Flops

Architectures Supported

<table>
<thead>
<tr>
<th>OFD, OFD4, OFD8, OFD16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFD, OFD4, OFD8, and OFD16 are single and multiple output D flip-flops.

The outputs (for example, Q3 – Q0) are connected to OPADs or IOPADs. The data on the D inputs is loaded into the flip-flops during the Low-to-High clock (C) transition and appears on the Q outputs.

The flip-flops are asynchronously cleared with Low outputs when power is applied, or when global reset is active.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

OFD Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFD Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

OFD Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
OFD8 Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
OFTD8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFD, you would infer an FD and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFD_1

Output D Flip-Flop with Inverted Clock

Architectures Supported

<table>
<thead>
<tr>
<th>OFD_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFD_1 is located in an input/output block (IOB). The output (Q) of the D flip-flop is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition and appears on the Q output.

The flip-flop is asynchronously cleared, output Low, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>↓</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

OFD_1 Implementation Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFD_1, you would infer an FD_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDDRCPE

Dual Data Rate Output D Flip-Flop with Clock Enable and Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>OFDDRCPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OFDDRCPE is a dual data rate (DDR) output D flip-flop with clock enable (CE) and asynchronous preset (PRE) and clear (CLR). It consists of one output buffer and one dual data rate flip-flop (FDDRCPE).

When the asynchronous PRE is High and CLR is Low, the Q output is preset High. When CLR is High, Q is set Low. Data on the D0 input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High C1 clock transition.

The INIT attribute does not apply to OFDDRCPE components.

The flip-flops are asynchronously cleared with Low outputs when power is applied.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>C1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
</tbody>
</table>

OFDDRCPE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

For HDL, the OFDDRCPE design element should be instantiated rather than inferred.

VHDL Instantiation Template

-- Component Declaration for OFDDRCPE should be placed
-- after architecture statement but before begin keyword

component OFDDRCPE
  port (Q : out STD_ULOGIC;
       C0 : in STD_ULOGIC;
       C1 : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       CLR : in STD_ULOGIC;
       D0 : in STD_ULOGIC;
       D1 : in STD_ULOGIC;
       PRE : in STD_ULOGIC);
end component;

-- Component Attribute specification for OFDDRCPE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OFDDRCPE should be placed
-- in architecture after the begin keyword

OFDDRCPE_INSTANCE_NAME : OFDDRCPE
  port map (Q => user_O,
           C0 => user_C0,
           C1 => user_C1,
           CE => user_CE,
           CLR => user_CLR,
           D0 => user_D0,
           D1 => user_D1,
           PRE => user_PRE);

Verilog Instantiation Template

OFDDRCPE_instance_name (.Q (user_O),
                         .C0 (user_C0),
                         .C1 (user_C1),
                         .CE (user_CE),
                         .CLR (user_CLR),
                         .D0 (user_D0),
                         .D1 (user_D1),
                         .PRE (user_I));
OFDDRRSE

Dual Data Rate Output D Flip-Flop with Synchronous Reset and Set and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>OFDDRRSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OFDDRRSE is a dual data rate (DDR) output D flip-flop with synchronous reset (R) and set (S) and clock enable (CE). It consists of one output buffer and one dual data rate flip-flop (FDDRRSE).

On a Low-to-High clock transition (C0 or C1), a High R input resets the Q output Low; a Low R input with a High S input sets Q High. When both R and S are Low and clock enable is High, data on the D0 input is loaded into the flip-flop on a Low-to-High C0 clock transition and data on the D1 input is loaded into the flip-flop on a Low-to-High C1 clock transition.

The flip-flops are asynchronously cleared with Low outputs when power is applied, or when global reset is active.

The INIT attribute does not apply to OFDDRRSE components.
OFDDRRSE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

For HDL, the OFDDRRSE design element should be instantiated rather than inferred.

VHDL Instantiation Template

```
-- Component Declaration for OFDDRRSE should be placed
-- after architecture statement but before begin keyword

component OFDDRRSE
  port (Q : out STD_ULOGIC;
       C0 : in STD_ULOGIC;
       C1 : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       D0 : in STD_ULOGIC;
       D1 : in STD_ULOGIC;
       R : in STD_ULOGIC;
       S : in STD_ULOGIC);
end component;

-- Component Attribute specification for OFDDRRSE
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for OFDDRRSE should be placed
-- in architecture after the begin keyword

OFDDRRSE_INSTANCE_NAME : OFDDRRSE
  port map (Q => user_Q,
           C0 => user_C0,
           C1 => user_C1,
           CE => user_CE,
           D0 => user_D0,
           D1 => user_D1,
           R => user_R,
           S => user_S);
```
Verilog Instantiation Template

OFDDRRSE instance_name (.Q (user_Q),
    .C0 (user_C0),
    .C1 (user_C1),
    .CE (user_CE),
    .D0 (user_D0),
    .D1 (user_D1),
    .R (user_R),
    .S (user_S));
Dual Data Rate D Flip-Flop with Active-Low 3-State Output Buffer, Clock Enable, and Asynchronous Preset and Clear

Architectures Supported

<table>
<thead>
<tr>
<th>OFDDRTCPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDDRTCPE is a dual data rate (DDR) D flip-flop with clock enable (CE) and asynchronous preset and clear whose output is enabled by a 3-state buffer. It consists of a dual data rate flip-flop (FDRCPE) and a 3-state output buffer (OBUFT). The data output (O) of the flip-flop is connected to the input of the output buffer (OBUFT). The output of the OBUFT is connected to an OPAD or IOPAD.

When the active-Low enable input (T) is Low, output is enabled and the data on the flip-flop’s Q output appears on the OBUFT’s O output. When the asynchronous PRE is High and CLR is Low, the O output is preset High. When CLR is High, O is set Low. Data on the D0 input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when PRE and CLR are Low and CE is High on the Low-to-High C1 clock transition.

When T is High, outputs are high impedance (Off). When CE is Low and T is Low, the outputs do not change.

The flip-flops are asynchronously cleared with Low outputs when power is applied.

The INIT attribute does not apply to OFDDRTCPE components.
OFDDRTCPE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

For HDL, the OFDDRTCPE design element should be instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for OFDDRTCPE should be placed
-- after architecture statement but before begin keyword
component OFDDRTCPE
    port (O : out STD_ULOGIC;
         C0 : in STD_ULOGIC;
         C1 : in STD_ULOGIC;
         CE : in STD_ULOGIC;
         CLR : in STD_ULOGIC;
         D0 : in STD_ULOGIC;
         D1 : in STD_ULOGIC;
         PRE : in STD_ULOGIC;
         T : in STD_ULOGIC);
end component;

-- Component Attribute specification for OFDDRTCPE
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes here

-- Component Instantiation for OFDDRTCPE should be placed
-- in architecture after the begin keyword

OFDDRTCPE_INSTANCE_NAME : OFDDRTCPE
    port map (O => user_O,
              C0 => user_C0,
              C1 => user_C1,
              CE => user_CE,
              CLR => user_CLR,
              D0 => user_D0,
              D1 => user_D1,
              PRE => user_PRE,
              T => user_T);
```
Verilog Instantiation Template

OFDDRTCPE instance_name (.O (user_0),
    .C0 (user_C0),
    .C1 (user_C1),
    .CE (user_CE),
    .CLR (user_CLR),
    .D0 (user_D0),
    .D1 (user_D1),
    .PRE (user_PRE),
    .T (user_T));
OFDDRTRESE

Dual Data Rate D Flip-Flop with Active-Low 3-State Output Buffer, Synchronous Reset and Set, and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OFDDRTRESE is a dual data rate (DDR) D flip-flop with clock enable (CE) and synchronous reset and set whose output is enabled by a 3-state buffer. It consists of a dual data rate flip-flop (FDDRTRESE) and a 3-state output buffer (OBUFT). The data output (O) of the flip-flop is connected to the input of the output buffer (OBUFT). The output of the OBUFT is connected to an OPAD or IOPAD.

When the active-Low enable input (T) is Low, output is enabled and the data on the flip-flop’s Q output appears on the OBUFT’s O output. On a Low-to-High clock transition (C0 or C1), a High R input resets the Q output Low; a Low R input with a High S input sets O High. When both R and S are Low and clock enable is High, data on the D0 input is loaded into the flip-flop on a Low-to-High C0 clock transition and data on the D1 input is loaded into the flip-flop on a Low-to-High C1 clock transition.

When T is High, outputs are high impedance (Off). When CE is Low and T is Low, the outputs do not change.

The flip-flops are asynchronously cleared with Low outputs when power is applied. The INIT attribute does not apply to OFDDRTRESE components.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>C1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
OFDDRTRSE Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage
For HDL, the OFDDRTRSE design element should be instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for OFDDRTRSE should be placed
-- after architecture statement but before begin keyword

component OFDDRTRSE
  port (O : out STD_ULOGIC;
       C0 : in STD_ULOGIC;
       C1 : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       D0 : in STD_ULOGIC;
       D1 : in STD_ULOGIC;
       R : in STD_ULOGIC;
       S : in STD_ULOGIC;
       T : in STD_ULOGIC);
end component;

-- Component Attribute specification for OFDDRTRSE
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes here

-- Component Instantiation for OFDDRTRSE should be placed
-- in architecture after the begin keyword

OFDDRTRSE_INSTANCE_NAME : OFDDRTRSE
  port map (O => user_O,
            C0 => user_C0,
            C1 => user_C1,
            CE => user_CE,
            D0 => user_D0,
```

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>C1</td>
</tr>
<tr>
<td>↑</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>↑</td>
</tr>
</tbody>
</table>
D1 => user_D1,
R => user_R,
S => user_S,
T => user_T);

Verilog Instantiation Template

OFDDRTRSE instance_name (.O (user_O),
  .C0 (user_C0),
  .C1 (user_C1),
  .CE (user_CE),
  .D0 (user_D0),
  .D1 (user_D1),
  .R (user_R),
  .S (user_S),
  .T (user_T));
OFDE, 4, 8, 16

D Flip-Flops with Active-High Enable Output Buffers

Architectures Supported

<table>
<thead>
<tr>
<th>OFDE, OFDE4, OFDE8, OFDE16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>No</td>
</tr>
</tbody>
</table>

OFDE, OFDE4, OFDE8, and OFDE16 are single or multiple D flip-flops whose outputs are enabled by 3-state buffers. The flip-flop data outputs (Q) are connected to the inputs of output buffers (OBUFE). The OBUFE outputs (O) are connected to OPADs or IOPADs. The data on the data inputs (D) is loaded into the flip-flops during the Low-to-High clock (C) transition. When the active-High enable inputs (E) are High, the data on the flip-flop outputs (Q) appears on the O outputs. When E is Low, outputs are high impedance (Z state or Off).

The flip-flops are asynchronously cleared with Low outputs when power is applied, or when global reset is active.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

OFDE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDE, you would infer an FDE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr o to pack all output registers into the IOBs.
OFDE_1

D Flip-Flop with Active-High Enable Output Buffer and Inverted Clock

Architectures Supported

<table>
<thead>
<tr>
<th>OFDE_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OFDE_1 and its output buffer are located in an input/output block (IOB). The data output of the flip-flop (Q) is connected to the input of an output buffer or OBUFE. The output of the OBUFE is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the High-to-Low clock (C) transition. When the active-High enable input (E) is High, the data on the flip-flop output (Q) appears on the O output. When E is Low, the output is high impedance (Z state or Off).

The flip-flop is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

OFDE_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFDE_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDE_1, you would infer an FDE_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDI

Output D Flip-Flop (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>OFDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDI is contained in an input/output block (IOB). The output (Q) of the D flip-flop is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q).

The flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>Q</td>
</tr>
</tbody>
</table>

OFDI Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFDI Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDI, you would infer an FDP and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr o to pack all output registers into the IOBs.
Output D Flip-Flop with Inverted Clock (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>OFDI_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDI_1 exists in an input/output block (IOB). The D flip-flop output (Q) is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition and appears on the Q output.

The flip-flop is asynchronously preset, output High, when power is applied.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>↓</td>
</tr>
</tbody>
</table>

OFDI_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFDI_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDI_1, you would infer an FDP_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDT, 4, 8, 16

Single and Multiple D Flip-Flops with Active-Low 3-State Output Enable Buffers

Architectures Supported

<table>
<thead>
<tr>
<th>OFDT, OFDT4, OFDT8, OFDT16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDT, OFDT4, OFDT8, and OFDT16 are single or multiple D flip-flops whose outputs are enabled by a 3-state buffers. The data outputs (Q) of the flip-flops are connected to the inputs of output buffers (OBUFT). The outputs of the OBUFTs (O) are connected to OPADs or IOPADs. The data on the data inputs (D) is loaded into the flip-flops during the Low-to-High clock (C) transition. When the active-Low enable inputs (T) are Low, the data on the flip-flop outputs (Q) appears on the O outputs. When T is High, outputs are high impedance (Off).

The flip-flops are asynchronously cleared with Low outputs, when power is applied, or when global reset is active.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>D</td>
</tr>
</tbody>
</table>

Libraries Guide
ISE 6.li
OFDT Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

OFDT Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

OFDT Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDT, you would infer an FDCE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDT_1

D Flip-Flop with Active-Low 3-State Output Buffer and Inverted Clock

Architectures Supported

<table>
<thead>
<tr>
<th>OFDT_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDT_1 and its output buffer are located in an input/output block (IOB). The flip-flop data output (Q) is connected to the input of an output buffer (OBUFT). The OBUFT output is connected to an OPAD or an IOPAD. The data on the data input (D) is loaded into the flip-flop on the High-to-Low clock (C) transition. When the active-Low enable input (T) is Low, the data on the flip-flop output (Q) appears on the O output. When T is High, the output is high impedance (Off).

The flip-flop is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

OFDT_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFDT_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDT_1, you would infer an FDCE_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pr o to pack all output registers into the IOBs.
Single- and Multiple-Output D Flip-Flops with Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>OFDX, OFDX4, OFDX8, OFDX16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro,</td>
</tr>
<tr>
<td>Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDX, OFDX4, OFDX8, and OFDX16 are single and multiple output D flip-flops. The Q outputs are connected to OPADs or IOPADs. The data on the D inputs is loaded into the flip-flops during the Low-to-High clock (C) transition and appears on the Q outputs. When CE is Low, flip-flop outputs do not change.

The flip-flops are asynchronously cleared with Low outputs, when power is applied, or when global reset is active.

Spartan-II, Spartan-IIIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>Dn</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

OFDX Implementation Spartan-II, Spartan-IIIIE, Virtex, Virtex-E
OFDX Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

OFDX8 Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-II Pro X
Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDX, you would infer an FDCE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDX_1

Output D Flip-Flop with Inverted Clock and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>OFDX_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDX_1 is located in an input/output block (IOB). The output (Q) of the D flip-flop is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition and appears on the Q output. When the CE pin is Low, the output (Q) does not change.

The flip-flop is asynchronously cleared with Low output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEx, or STARTUP_VIRTEx2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

OFDX_1 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFDX_1 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDX_1, you would infer an FDCE_1 and put the IOB=TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDXI

Output D Flip-Flop with Clock Enable (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>OFDXI</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

OFDXI is contained in an input/output block (IOB). The output (Q) of the D flip-flop is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). When CE is Low, the output does not change.

The flip-flop is asynchronously preset with High output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

OFDXI Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
OFDXI Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDXI, you would infer an FDPE and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
OFDXI_1

Output D Flip-Flop with Inverted Clock and Clock Enable (Asynchronous Preset)

Architectures Supported

<table>
<thead>
<tr>
<th>OFDXI_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

OFDXI_1 is located in an input/output block (IOB). The D flip-flop output (Q) is connected to an OPAD or an IOPAD. The data on the D input is loaded into the flip-flop during the High-to-Low clock (C) transition and appears on the Q output. When CE is Low, the output (Q) does not change.

The flip-flop is asynchronously preset with High output when power is applied, or when global reset is active.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

OFDXI_1 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
Usage

This component is inside of the IOB. It cannot be directly inferred. The most common design practice is to infer a regular component and put an IOB=TRUE attribute on the component in the UCF file or in the code. For instance, to get an OFDXI_1, you would infer an FDPE_1 and put the IOB = TRUE attribute on the component. Or, you could use the map option –pro to pack all output registers into the IOBs.
Single- and Multiple-Output Pads

Architectures Supported

| OPAD       | Spartan-II, Spartan-IIIE | Primitive
|------------|--------------------------|-----------
|            | Spartan-3                | Primitive
|            | Virtex, Virtex-E         | Primitive
|            | Virtex-II, Virtex-II Pro, Virtex-II Pro X | Primitive
|            | XC9500, XC9500XV, XC9500XL | Primitive
|            | CoolRunner XPLA3         | Primitive
|            | CoolRunner-II            | Primitive
| OPAD4, OPAD8, OPAD16 | Spartan-II, Spartan-IIIE | Macro
|            | Spartan-3                | Macro
|            | Virtex, Virtex-E         | Macro
|            | Virtex-II, Virtex-II Pro, Virtex-II Pro X | Macro
|            | XC9500, XC9500XV, XC9500XL | Macro
|            | CoolRunner XPLA3         | Macro
|            | CoolRunner-II            | Macro

OPAD, OPAD4, OPAD8, and OPAD16 are single and multiple output pads. An OPAD connects a device pin to an output signal of a PLD. It is internally connected to an input/output block (IOB), which is configured by the software as an OBUF, an OBUFT, an OBUFE, an OFD, or an OFDT.

See the appropriate CAE tool interface user guide for details on assigning pin location and identification.

Usage

For HDL, it is not necessary to use these elements in the design. They will be added automatically.

Commonly Used Constraints

IOBDELAY, PULLDOWN
# OR2-9

2- to 9-Input OR Gates with Inverted and Non-Inverted Inputs

<table>
<thead>
<tr>
<th>OR2, OR2B1, OR2B2, OR3, OR3B1, OR3B2, OR3B3, OR4, OR4B1, OR4B2, OR4B3, OR4B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR5, OR5B1, OR5B2, OR5B3, OR5B4, OR5B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II PRO</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR6, OR7, OR8, OR9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II PRO</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>
OR Gate Representations
The OR function is performed in the Configurable Logic Block (CLB) function generators for Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X. OR functions of up to five inputs are available in any combination of inverting and non-inverting inputs. OR functions of six to nine inputs are available with only non-inverting inputs. To invert some or all inputs, use external inverters. Since each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

See “OR2-9” for information on additional OR functions for the Spartan-II, Spartan-IIE, Virtex, and Virtex-E.
**Usage**

OR2 through OR5 are primitives that can be inferred or instantiated. OR6 through OR9 are macros which can be inferred.

**VHDL Inference Code**

```vhdl
architecture Behavioral of or6 is
begin
  process (I0, I1, I2, I3, I4, I5)
  begin
    O <= (I0 or I1 or I2 or I3 or I4 or I5);
  end process;
end Behavioral;
```

**Verilog Inference Code (OR6)**

```verilog
always @(I0 or I1 or I2 or I3 or I4 or I5)
begin
  O <= (I0 || I1 || I2 || I3 || I4 || I5);
end
```

**VHDL Instantiation Template for OR5**

Following is the VHDL code for OR5. To instantiate OR2, remove I2, I3, and I4. To instantiate OR3, remove I3 and I4. For OR4, remove I4. OR2B1, and OR2B2 have the same code as OR2. OR3B1, 3B2, and 3B3 have the same code as OR3 and so forth.
-- Component Declaration for OR5 should be placed
-- after architecture statement but before begin keyword

component OR5
  port (O : out STD_ULOGIC;
       I0 : in STD_ULOGIC;
       I1 : in STD_ULOGIC;
       I2 : in STD_ULOGIC;
       I3 : in STD_ULOGIC;
       I4: in STD_ULOGIC);
end component;

-- Component Attribute specification for OR5
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for OR5 should be placed
-- in architecture after the begin keyword

OR5_INSTANCE_NAME : OR5
  port map (O => user_O,
            I0 => user_I0,
            I1 => user_I1,
            I2 => user_I2,
            I3 => user_I3,
            I4 => user_I4);

Verilog Instantiation Template for OR5

OR5 OR5_instance_name (.O (user_O),
                       .I0 (user_I0),
                       .I1 (user_I1),
                       .I2 (user_I2),
                       .I3 (user_I3),
                       .I4 (user_I4));

Commonly Used Constraints

None
OR12, 16

12- and 16-Input OR Gates with Non-Inverted Inputs

Architectures Supported

<table>
<thead>
<tr>
<th>OR12, OR16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II PRO</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

See “OR2-9” for information on OR functions.
OR16 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E
OR16 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

Usage

For HDL, OR12 and OR16 are macros that are inferred. See “OR2-9” for information about inferring OR gates.
ORCY

OR with Carry Logic

Architectures Supported

<table>
<thead>
<tr>
<th>ORCY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ORCY is a special OR with general O output used for generating faster and smaller arithmetic functions.

Each Virtex-II, Virtex-II Pro, and Virtex-II Pro X slice contains a dedicated 2-input OR gate that ORs together carry out values for a series of horizontally adjacent carry chains. The OR gate gets one input external to the slice and the other input from the output of the high order carry mux. The OR gate’s output drives the next slice’s OR gate horizontally across the die.

Only MUXCY outputs can drive the signal on the CI pin. Only ORCY outputs or logic zero can drive the I pin.

Usage

For HDL, the ORCY design element should be instantiated rather than inferred.

VHDL Instantiation Template

-- Component Declaration for ORCY should be placed
-- after architecture statement but before begin keyword

component ORCY
    port (O : out STD_ULOGIC;
         CI : in STD_ULOGIC;
         I : in STD_ULOGIC);
end component;

-- Component Attribute specification for ORCY
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for ORCY should be placed
-- in architecture after the begin keyword

ORCY_INSTANCE_NAME : ORCY
port map (O => user_O,
    CI => user_CI,
    I => user_I);

Verilog Instantiation Template

ORCY instance_name (.O (user_O),
    .CI (user_CI),
    .I (user_I));

Commonly Used Constraints

None
The PowerPC 405 embedded core is a 32-bit RISC core integrating a PowerPC 405 CPU, separate instruction and data caches, a JTAG port, trace FIFO, multiple timers, and a memory management unit (MMU). Integrated on-chip memory (OCM) controllers provide dedicated interfaces between Block SelectRAM memory and the processor core instruction and data paths for high-speed access. The PowerPC 405 core implements the PowerPC User Instruction Set.

For complete information about the PowerPC 405, see the following documents:

- Virtex-II Pro Datasheet
- Virtex-II Pro Handbook
- The PowerPC 405 Core Processor Block Manual
- The PowerPC 405 User Guide

The following table lists the inputs and outputs of the primitive. For detailed information about the pinouts, see the DS083 Virtex-II Pro Data Sheet.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAMDSOCMCLK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>BRAMDSOCMRRDDBUS [0:31]</td>
<td>C405CPMMSRCE</td>
</tr>
<tr>
<td>BRAMISOCMCLK</td>
<td>C405CPMSREE</td>
</tr>
<tr>
<td>BRAMISOCMRRDDBUS [0:63]</td>
<td>C405CPMTIMERIRQ</td>
</tr>
<tr>
<td>CPMC405CLOCK</td>
<td>C405CPMTIMERRESETREQ</td>
</tr>
<tr>
<td>CPMC405CORECLKINACTIVE</td>
<td>C405DBGMSRWE</td>
</tr>
<tr>
<td>CPMC405CPUCLKEN</td>
<td>C405DBGSTOPACK</td>
</tr>
<tr>
<td>CPMC405JTAGCLKEN</td>
<td>C405DBGWBCOMPLETE</td>
</tr>
<tr>
<td>CPMC405TIMERCLKEN</td>
<td>C405DBGWBFULL</td>
</tr>
<tr>
<td>CPMC405TIMERTICK</td>
<td>C405DBGWBIAR[0:29]</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>DBGC405DEBUGHALT</td>
<td>C405DCRBUS [0:9]</td>
</tr>
<tr>
<td>DBGC405EXTBUSHOSTACK</td>
<td>C405DCRDBUSOUT [0:31]</td>
</tr>
<tr>
<td>DBGC405UNCONDDEBUGEVENT</td>
<td>C405DCREAD</td>
</tr>
<tr>
<td>DCRC405ACK</td>
<td>C405DCRWRITE</td>
</tr>
<tr>
<td>DCRC405DBUSIN [0:31]</td>
<td>C405JTGCAUGHTEDR</td>
</tr>
<tr>
<td>DSARCVVALUE [0:7]</td>
<td>C405JTGEXIT</td>
</tr>
<tr>
<td>DSCNTLVALUE [0:7]</td>
<td>C405JTPGMOUT</td>
</tr>
<tr>
<td>EICC405CRITINPUTIRQ</td>
<td>C405JTGSHIFTDR</td>
</tr>
<tr>
<td>EICC405EXTINPUTIRQ</td>
<td>C405JTGTDN</td>
</tr>
<tr>
<td>ISARCVVALUE [0:7]</td>
<td>C405JTGDOEN</td>
</tr>
<tr>
<td>ISCNTLVALUE [0:7]</td>
<td>C405JGTUPEDR</td>
</tr>
<tr>
<td>JTG405BNDSCANTDO</td>
<td>C405PLBDCUABORT</td>
</tr>
<tr>
<td>JTG405TCK</td>
<td>C405PLBDCUABUS [0:31]</td>
</tr>
<tr>
<td>JTG405TDI</td>
<td>C405PLBDCUBE [0:7]</td>
</tr>
<tr>
<td>JTG405TMS</td>
<td>C405PLBDCUCACHEABLE</td>
</tr>
<tr>
<td>JTG405TRSTNEG</td>
<td>C405PLBDCUGUARDED</td>
</tr>
<tr>
<td>MCBCPUCLKEN</td>
<td>C405PLBDCUPRIORITY [0:1]</td>
</tr>
<tr>
<td>MCBJTAGEN</td>
<td>C405PLBDCUREQUEST</td>
</tr>
<tr>
<td>MCBTIMEMEN</td>
<td>C405PLBDCUWRNW</td>
</tr>
<tr>
<td>MCPPCRST</td>
<td>C405PLBDCUSIZE2</td>
</tr>
<tr>
<td>PLBC405DCUADDRACK</td>
<td>C405PLBDCU0ATTR</td>
</tr>
<tr>
<td>PLBC405DCUDBUSY</td>
<td>C405PLBDCUWRDBUS [0:63]</td>
</tr>
<tr>
<td>PLBC405DCUERR</td>
<td>C405PLBDCUWRITETHRU</td>
</tr>
<tr>
<td>PLBC405DCURDDACK</td>
<td>C405PLBDCUABORT</td>
</tr>
<tr>
<td>PLBC405DCURDDBUS [0:63]</td>
<td>C405PLBDCUABUS [0:29]</td>
</tr>
<tr>
<td>PLBC405DCURDDWADDR [1:3]</td>
<td>C405PLBDCUCACHEABLE</td>
</tr>
<tr>
<td>PLBC405DCUSIZE1</td>
<td>C405PLBDCUPRIORITY [0:1]</td>
</tr>
<tr>
<td>PLBC405DCUWRDDACK</td>
<td>C405PLBDCUREQUEST</td>
</tr>
<tr>
<td>PLBC405ICUADDRACK</td>
<td>C405PLBDCUSIZE2 [2:3]</td>
</tr>
<tr>
<td>PLBC405ICUDBUSY</td>
<td>C405PLBDCU0ATTR</td>
</tr>
<tr>
<td>PLBC405ICUERR</td>
<td>C405RSTCHIPRESETREQ</td>
</tr>
<tr>
<td>PLBC405ICURDDACK</td>
<td>C405RSTCORERESETREQ</td>
</tr>
<tr>
<td>PLBC405ICURDDBUS [0:63]</td>
<td>C405RSTSRESETREQ</td>
</tr>
<tr>
<td>PLBC405ICURDDWADDR [1:3]</td>
<td>C405TRCCYCLE</td>
</tr>
<tr>
<td>PLBC405ICUSSIZE1</td>
<td>C405TRCEVENEXECUTIONSTATUS [0:1]</td>
</tr>
<tr>
<td>PLBCLK</td>
<td>C405TRCODDEXECUTIONSTATUS [0:1]</td>
</tr>
<tr>
<td>RSTC405RESETCHIP</td>
<td>C405TRCTRACESTATUS [0:3]</td>
</tr>
<tr>
<td>RSTC405RESETCORE</td>
<td>C405TRCTRIGGEREVENTOUT</td>
</tr>
<tr>
<td>RSTC405RESETSYS</td>
<td>C405TRCTRIGGEREVENTTYPE [0:10]</td>
</tr>
<tr>
<td>TIEC405DETERMINISTICMULT</td>
<td>C405XXXMACHINECHECK</td>
</tr>
<tr>
<td>TIEC405DISOPERANDFWD</td>
<td>DSOCMBRAMABUS [8:29]</td>
</tr>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>TIEC405MMUEN</td>
<td>DSOCMBRAMBYTEWRITE [0:3]</td>
</tr>
<tr>
<td>TIEDSOCMDCRADDR [0:7]</td>
<td>DSOCMBRAMEN</td>
</tr>
<tr>
<td>TIEISOCMDCRADDR [0:7]</td>
<td>DSOCMBRAMWRDBUS [0:31]</td>
</tr>
<tr>
<td>TRCC405TRACEDISABLE</td>
<td>DSOCMBUSY</td>
</tr>
<tr>
<td>TRCC405TRIGGEREVENTIN</td>
<td>ISOCMBRAMEN</td>
</tr>
<tr>
<td></td>
<td>ISOCMBRAMEVENWRITEEN</td>
</tr>
<tr>
<td></td>
<td>ISOCMBRAMODDDWRITEEN</td>
</tr>
<tr>
<td></td>
<td>ISOCMBRAMRDABUS [8:28]</td>
</tr>
<tr>
<td></td>
<td>ISOCMBRAMWRABUS [8:28]</td>
</tr>
<tr>
<td></td>
<td>ISOCMBRAMWRDBUS [0:31]</td>
</tr>
<tr>
<td>Signal Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>PPC405</td>
<td></td>
</tr>
<tr>
<td>BRAMDSOCMCLK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>BRAMDSOCM RDDBUS(0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>BRAMDSOCMCLK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>issocmbramwrdbus(0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>CPM0C405CLK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>CPM0C405CORECLKINACTIVE</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>CPM0C405CLOCK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>CPM0C405TIAGCLK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>CPM0C405MERTICK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>DBG0C405DEVJ OGALT</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>DBG0C405EXTBUS HOLDACK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>DBG0C405UNC3ND DEBEUG EVENT</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>DCRC0C405ACK</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>DCRC0C405DBUS (0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>ISOCMBRAMEVENWRITE</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>ISOCMBRAMEVENWRITE</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>ISOCMBRAMWREDFEWRITTEN</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>ISOCMBRAMEVENWRITE</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>issocmbramwrdbus(0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>issocmbramwrdbus(0:28)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>issocmbramwrdbus(0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>issocmbramwrdbus(0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
<tr>
<td>issocmbramwrdbus(0:31)</td>
<td>C405CPMCORESLEEPREQ</td>
</tr>
</tbody>
</table>
Usage

For HDL, the PPC405 design element is instantiated rather than inferred.

VHDL Instantiation Template

-- Component Declaration for PPC405 should be placed
-- after architecture statement but before begin keyword

component PPC405

port (C405CPMCORESLEEPREQ : out STD_ULOGIC;
      C405CPMMRSRCE      : out STD_ULOGIC;
      C405CPMMSREE       : out STD_ULOGIC;
      C405CPMTIMERIRQ    : out STD_ULOGIC;
      C405CPMTIMERRESETREQ: out STD_ULOGIC;
      C405DBGMSRWE       : out STD_ULOGIC;
      C405DBGSTOPACK     : out STD_ULOGIC;
      C405DBGWBCOMPLETE  : out STD_ULOGIC;
      C405DBGWBFULL      : out STD_ULOGIC;
      C405DBGWBIAR       : out STD_LOGIC_VECTOR (29 downto 0);
      C405DCRABUS        : out STD_LOGIC_VECTOR (31 downto 0);
      C405DCRREAD        : out STD_ULOGIC;
      C405DCRWRITE       : out STD_ULOGIC;
      C405JTGCAPTUREDR   : out STD_ULOGIC;
      C405JTGEXTEST      : out STD_ULOGIC;
      C405JTGPGMOUT      : out STD_ULOGIC;
      C405JTGSHIFTDR     : out STD_ULOGIC;
      C405JTGTDODE       : out STD_ULOGIC;
      C405JTGDDEPTH      : out STD_ULOGIC;
      C405PLBDUCUABORT   : out STD_ULOGIC;
      C405PLBDUCUABUS    : out STD_LOGIC_VECTOR (31 downto 0);
      C405PLBDCUBE       : out STD_LOGIC_VECTOR (7 downto 0);
      C405PLBDUCACHEABLE : out STD_ULOGIC;
      C405PLBDUCGUARDED  : out STD_ULOGIC;
      C405PLBDUPRIORITY  : out STD_LOGIC_VECTOR (1 downto 0);
C405PLBDCCUREQUEST : out STD_LOGIC;
C405PLBDCURNW : out STD_LOGIC;
C405PLBDCUSIZE2 : out STD_LOGIC;
C405PLBUCU0ATTR : out STD_LOGIC;
C405PLBDCUWRDBUS : out STD_LOGIC_VECTOR (63 downto 0);
C405PLBDCUWRITETHRU : out STD_LOGIC;
C405PLBICUABORT : out STD_LOGIC;
C405PLBICUABORT : out STD_LOGIC_VECTOR (29 downto 0);
C405PLBICUCACHEABLE : out STD_LOGIC;
C405PLBICUPRIORITY : out STD_LOGIC_VECTOR (1 downto 0);
C405PLBICUREQUEST : out STD_LOGIC;
C405PLBICUSIZE : out STD_LOGIC_VECTOR (3 downto 2);
C405PLBUCU0ATTR : out STD_LOGIC;
C405RSTCHIPRESETREQ : out STD_LOGIC;
C405RSTCORERESETREQ : out STD_LOGIC;
C405RSTSYSRESETREQ : out STD_LOGIC;
C405TRCCYCLE : out STD_LOGIC;
C405TRCEVENEXECUTIONSTATUS : out STD_LOGIC_VECTOR (1 downto 0);
C405TRCODDEXECUTIONSTATUS : out STD_LOGIC_VECTOR (1 downto 0);
C405TRCTRACESTATUS : out STD_LOGIC_VECTOR (3 downto 0);
C405TRCTRIGGEREVENTOUT : out STD_LOGIC;
C405TRCTRIGGEREVENTTYPE : out STD_LOGIC_VECTOR (10 downto 0);
C405XXXMACHINECHECK : out STD_LOGIC;
DSOCMBRAMABUS : out STD_LOGIC_VECTOR (29 downto 8);
DSOCMBRAMBYTEWRITE : out STD_LOGIC_VECTOR (3 downto 0);
DSOCMBRAMEN : out STD_LOGIC;
DSOCMBRAMWRDBUS : out STD_LOGIC_VECTOR (31 downto 0);
ISOCMBUSY : out STD_LOGIC;
ISOCMBRAMEN : out STD_LOGIC;
ISOCMBRAMMEVENWRITEEN : out STD_LOGIC;
ISOCMBRAMODDWRITEEN : out STD_LOGIC;
ISOCMBRAMRDABUS : out STD_LOGIC_VECTOR (28 downto 8);
ISOCMBRAMWRABUS : out STD_LOGIC_VECTOR (28 downto 8);
ISOCMBRAMWRDBUS : out STD_LOGIC_VECTOR (31 downto 0);
BRAMDSOCMCLK : in STD_LOGIC;
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAMDSOCRMRDDBUS</td>
<td>in STD_LOGIC_VECTOR (31 downto 0)</td>
</tr>
<tr>
<td>BRAMISOCMCLK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>BRAMISOCRMRDDBUS</td>
<td>in STD_LOGIC_VECTOR (63 downto 0)</td>
</tr>
<tr>
<td>CPMC405CLOCK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>CPMC405CORECLKINACTIVE</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>CPMC405CPUCLKEN</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>CPMC405JTAGCLKEN</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>CPMC405TIMERCLKEN</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>CPMC405TIMERTICK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>DBGC405DEBUGHALT</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>DBGC405EXTBUSHOLDACK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>DBGC405UNCONDEBUGEVENT</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>DCRC405ACK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>DCRC405DBUSIN</td>
<td>in STD_LOGIC_VECTOR (31 downto 0)</td>
</tr>
<tr>
<td>DSARCF405VALUE</td>
<td>in STD_LOGIC_VECTOR (7 downto 0)</td>
</tr>
<tr>
<td>DSCNTLVALUE</td>
<td>in STD_LOGIC_VECTOR (7 downto 0)</td>
</tr>
<tr>
<td>EICC405CRITINPUTIRQ</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>EICC405EXTINPUTIRQ</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>ISARCF405VALUE</td>
<td>in STD_LOGIC_VECTOR (7 downto 0)</td>
</tr>
<tr>
<td>ISCNTLVALUE</td>
<td>in STD_LOGIC_VECTOR (7 downto 0)</td>
</tr>
<tr>
<td>JTGC405BNDSCANTDO</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>JTGC405TCK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>JTGC405TDI</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>JTGC405TMS</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>JTGC405TRSTNEG</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>MCBCPUCLKEN</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>MCBJTAGEN</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>MCBTIMEREN</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>MCPPCRST</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>PLBC405DCUADDRACK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>PLBC405DCUBUSY</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>PLBC405DCUERR</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>PLBC405DCURDDACK</td>
<td>in STD_LOGIC;</td>
</tr>
<tr>
<td>PLBC405DCURDDDBUS</td>
<td>in STD_LOGIC_VECTOR (63 downto 0)</td>
</tr>
</tbody>
</table>
PLBC405DCURDWDADDR : in STD_LOGIC_VECTOR (3 downto 1);
PLBC405DCUSSIZE1 : in STD_ULOGIC;
PLBC405DCUWRDACK : in STD_ULOGIC;
PLBC405ICUADDRACK : in STD_ULOGIC;
PLBC405ICUBUSY : in STD_ULOGIC;
PLBC405ICUERR : in STD_ULOGIC;
PLBC405ICURDDACK : in STD_ULOGIC;
PLBC405ICURDDBUS : in STD_LOGIC_VECTOR (63 downto 0);
PLBC405ICURDWDADDR : in STD_LOGIC_VECTOR (3 downto 1);
PLBC405ICUSSIZE1 : in STD_ULOGIC;
PLBCLK : in STD_ULOGIC;
RSTC405RESETCHIP : in STD_ULOGIC;
RSTC405RESETCORE : in STD_ULOGIC;
RSTC405RESETSYS : in STD_ULOGIC;
TIEC405DETERMINISTICMULT : in STD_ULOGIC;
TIEC405DISOPERANDFWD : in STD_ULOGIC;
TIEC405MMUEN : in STD_ULOGIC;
TIEDSOCMDCRADDR : in STD_LOGIC_VECTOR (7 downto 0);
TIEISOCMDCRADDR : in STD_LOGIC_VECTOR (7 downto 0);
TRCC405TRACEDISABLE : in STD_ULOGIC;
TRCC405TRIGGEREVENTINE : in STD_ULOGIC);
end component;

-- Component Attribute specification for PPC405
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for PPC405 should be placed
-- in architecture after the begin keyword

PPC405_INSTANCE_NAME : PPC405
port map (C405CPMCORESLEEPRQ => user_C405CPMCORESLEEPRQ,
C405CPMMSRCE => user_C405CPMMSRCE,
C405CPMMSREE => user_C405CPMMSREE,
C405CPMTIMERIRQ => user_C405CPMTIMERIRQ,
C405CPMTIMERRESETREQ => user_C405CPMTIMERRESETREQ,
C405DBGMSRWEE => user_C405DBGMSRWEE,
C405DBGSTOPACK => user_C405DBGSTOPACK,
C405DBGWBCOMPLETE => user_C405DBGWBCOMPLETE,
C405DBGWBFULL => user_C405DBGWBFULL,
C405DBGWBIAR => user_C405DBGWBIAR,
C405DCRABUS => user_C405DCRABUS,
C405DCRDBUSOUT => user_C405DCRDBUSOUT,
C405DCRREAD => user_C405DCRREAD,
C405DCRWRIWE => user_C405DCRWRIWE,
C405JTGCAPTUREDR => user_C405JTGCAPTUREDR,
C405JTGEXTEST => user_C405JTGEXTEST,
C405JTGPGMOUT => user_C405JTGPGMOUT,
C405JTGSHIITDR => user_C405JTGSHIITDR,
C405JTGTDO => user_C405JTGTDO,
C405JTGTDOEN => user_C405JTGTDOEN,
C405JTGUPIATEDR => user_C405JTGUPIATEDR,
C405PLBDCUABORT => user_C405PLBDCUABORT,
C405PLBDCUABUS => user_C405PLBDCUABUS,
C405PLBDLCUBE => user_C405PLBDLCUBE,
C405PLBDUCACHEABLE => user_C405PLBDUCACHEABLE,
C405PLBDUCGUARDED => user_C405PLBDUCGUARDED,
C405PLBDUCUPRIORITY => user_C405PLBDUCUPRIORITY,
C405PLBDUCREQUEST => user_C405PLBDUCREQUEST,
C405PLBDUCURNW => user_C405PLBDUCURNW,
C405PLBDUCUSIZE2 => user_C405PLBDUCUSIZE2,
C405PLBDUCU0ATTR => user_C405PLBDUCU0ATTR,
C405PLBDCUWRDBUS => user_C405PLBDCUWRDBUS,
C405PLBDCUWRITETHRU => user_C405PLBDCUWRITETHRU,
C405PLBICUABORT => user_C405PLBICUABORT,
C405PLBICUABUS => user_C405PLBICUABUS,
C405PLBICUCACHEABLE => user_C405PLBICUCACHEABLE,  
C405PLBICUPRIORITY => user_C405PLBICUPRIORITY,  
C405PLBICUREQUEST => user_C405PLBICUREQUEST,  
C405PLBICUSIZE => user_C405PLBICUSIZE,  
C405PLBICUU0ATTR => user_C405PLBICUU0ATTR,  
C405RSTCHIPRESETREQ => user_C405RSTCHIPRESETREQ,  
C405RSTCORERESETREQ => user_C405RSTCORERESETREQ,  
C405RSTSYSRESETREQ => user_C405RSTSYSRESETREQ,  
C405TRCCYCLE => user_C405TRCCYCLE,  
C405TRCEVENEXECUTIONSTATUS => user_C405TRCEVENEXECUTIONSTATUS,  
C405TRCODDDEEXECUTIONSTATUS => user_C405TRCODDDEEXECUTIONSTATUS,  
C405TRCTRACESTATUS => user_C405TRCTRACESTATUS,  
C405TRCTRIGGEREVENTOUT => user_C405TRCTRIGGEREVENTOUT,  
C405TRCTRIGGEREVENTTYPE => user_C405TRCTRIGGEREVENTTYPE,  
C405XXXMACHINECHECK => user_C405XXXMACHINECHECK,  
DSOCMBRAMABUS => user_DSOCMBRAMABUS,  
DSOCMBRAMBYTEWRITE => user_DSOCMBRAMBYTEWRITE,  
DSOCMBRAMEN => user_DSOCMBRAMEN,  
DSOCMBRAMWRDBUS => user_DSOCMBRAMWRDBUS,  
DSOCMBUSY => user_DSOCMBUSY,  
ISOCMBRAMEN => user_ISOCMBRAMEN,  
ISOCMBRAMEVENWRITEEN => user_ISOCMBRAMEVENWRITEEN,  
ISOCMBRAMODDWRITEEN => user_ISOCMBRAMODDWRITEEN,  
ISOCMBRAMRDABUS => user_ISOCMBRAMRDABUS,  
ISOCMBRAMWRABUS => user_ISOCMBRAMWRABUS,  
ISOCMBRAMWRDBUS => user_ISOCMBRAMWRDBUS,  
BRAMDSOCMCLK => user_BRAMDSOCMCLK,  
BRAMDSOCMRDBUS => user_BRAMDSOCMRDBUS,  
BRAMISOCMCLK => user_BRAMISOCMCLK,  
BRAMISOCMRDBUS => user_BRAMISOCMRDBUS,  
CPMC405CLOCK => user_CPMC405CLOCK,  
CPMC405CORECLINKINACTIVE => user_CPMC405CORECLINKINACTIVE,  
CPMC405CPUCLKEN => user_CPMC405CPUCLKEN,  
CPMC405JTAGCLKEN => user_CPMC405JTAGCLKEN,
CPMC405TIMERCLEN => user_CPM C405TIMERCLEN,
CPMC405TIMERTICK => user_CPM C405TIMERTICK,
DBGC405DEBGHA LT => user_DBGC405DEBG HA LT,

DBGC405EXTBUSHoldAck => user_DBGC405EXTBUSHoldAck,
DBGC405UNCONDDEBUGEVENT => user_DBGC405UNCONDDEBUGEVENT,
DCRC405ACK => user_DCRC405ACK,
DCRC405DBUSIN => user_DCRC405DBUSIN,
DSARCVALUE => user_DSARCVALUE,
DSCNTLVALUE => user_DSCNTLVALUE,
EICC405CRITINPUTIRQ => user_EICC405CRITINPUTIRQ,
EICC405EXTINPUTIRQ => user_EICC405EXTINPUTIRQ,
ISARCVALUE => user_ISARCVALUE,
ISCNTLVALUE => user_ISCN TLVALUE,
JTGC405BNDSCANTDO => user_JTGC405BNDSCANTDO,
JTGC405TCK => user_JTGC405TCK,
JTGC405TDI => user_JTGC405TDI,
JTGC405TMS => user_JTGC405TMS,
JTGC405TRSTNEG => user_JTGC405TRSTNEG,
MCBCPUCLKEN => user_MCBCPUCLKEN,
MCBJTAGEN => user_MCBJTAGEN,
MCBTIMEREN => user_MCBTIMEREN,
MCPPCRST => user_MCPPCRST,
PLBC405DCUADDRACK => user_PLBC405DCUADDRACK,
PLBC405DCUBUSY => user_PLBC405DCUBUSY,
PLBC405DCUERR => user_PLBC405DCUERR,
PLBC405DCURDDACK => user_PLBC405DCURDDACK,
PLBC405DCURDDBUS => user_PLBC405DCURDDBUS,
PLBC405DCURDWADDR => user_PLBC405DCURDWADDR,
PLBC405DCUSSIZE1 => user_PLBC405DCUSSIZE1,
PLBC405DCUWRDACK => user_PLBC405DCUWRDACK,
PLBC405ICUADDRACK => user_PLBC405ICUADDRACK,
PLBC405ICUBUSY => user_PLBC405ICUBUSY,
PLBC405ICUERR => user_PLBC405ICUERR,
PLBC405ICURDDACK => user_PLBC405ICURDDACK,
Verilog Instantiation Template

PPC405 instance_name (.C405CPMCORESLEEPREQ
(user_C405CPMCORESLEEPREQ),
   .C405CPMMRSCE (user_C405CPMMRSCE),
   .C405CPMMRSRE (user_C405CPMMRSRE),
   .C405CPMRSIRIQ (user_C405CPMRSIRIQ),
   .C405CPMMRSRESETREQ (user_C405CPMRSRESETREQ),
   .C405DBGMSRWE (user_C405DBGMSRWE),
   .C405DBGSTOPACK (user_C405DBGSTOPACK),
   .C405DBGWBCOMPLETE (user_C405DBGWBCOMPLETE),
   .C405DBGWBFULL (user_C405DBGWBFULL),
   .C405DBGWBIAR (user_C405DBGWBIAR),
   .C405DCRABUS (user_C405DCRABUS),
   .C405DCRDBUSOUT (user_C405DCRDBUSOUT),
   .C405DCRREAD (user_C405DCRREAD),
   .C405DCRWRITE (user_C405DCRWRITE),
   .C405JTGCAPTUREDR (user_C405JTGCAPTUREDR),
   .C405JTGEXTEST (user_C405JTGEXTEST),
   .C405JTGPGMOUT (user_C405JTGPGMOUT),
   .C405JTGHIFDR (user_C405JTGHIFDR),
   .C405JTGTDO (user_C405JTGTDO),
.C405JTGTDOEN (user_C405JTGTDOEN),
.C405JTGUPTEDR (user_C405JTGUPTEDR),
.C405PLBDCUABORT (user_C405PLBDCUABORT),
.C405PLBDCUABUS (user_C405PLBDCUABUS),
.C405PLBDCUBE (user_C405PLBDCUBE),
.C405PLBDCUCACHEABLE (user_C405PLBDCUCACHEABLE),
.C405PLBDCUGUARDED (user_C405PLBDCUGUARDED),
.C405PLBDCUPRIORITY (user_C405PLBDCUPRIORITY),
.C405PLBDCUREQUEST (user_C405PLBDCUREQUEST),
.C405PLDCURNW (user_C405PLDCURNW),
.C405PLBDCUSIZE2 (user_C405PLBDCUSIZE2),
.C405PLBDCU0ATTR (user_C405PLBDCU0ATTR),
.C405PLBDCUWRDBUS (user_C405PLBDCUWRDBUS),
.C405PLBDCUWRITETHRU (user_C405PLBDCUWRITETHRU),
.C405PLBICUABORT (user_C405PLBICUABORT),
.C405PLBICUABUS (user_C405PLBICUABUS),
.C405PLBICUCACHEABLE (user_C405PLBICUCACHEABLE),
.C405PLBICUPRIORITY (user_C405PLBICUPRIORITY),
.C405PLBICUREQUEST (user_C405PLBICUREQUEST),
.C405PLBICUSIZE (user_C405PLBICUSIZE),
.C405PLBICU0ATTR (user_C405PLBICU0ATTR),
.C405RSTCHIPRESETREQ (user_C405RSTCHIPRESETREQ),
.C405RSTCORERESETREQ (user_C405RSTCORERESETREQ),
.C405RSTSYSRESETREQ (user_C405RSTSYSRESETREQ),
.C405TRCCYCLE (user_C405TRCCYCLE),
.C405TRCEVENEXECUTIONSTATUS(user_C405TRCEVENEXECUTIO
NSTATUS),
.C405TRCDDEEXECUTIONSTATUS (user_C405TRCDDEEXECUTIONSTATUS),
.C405TRCTRACESTATUS (user_C405TRCTRACESTATUS),
.C405TRCTRIGGEREVENTOUT (user_C405TRCTRIGGEREVENTOUT),
.C405TRCTRIGGEREVENTTYPE (user_C405TRCTRIGGEREVENTTYPE),
.C405XXXMACHINECHECK (user_C405XXXMACHINECHECK),
.DSOCMBRAMABUS (user_DSOCMBRAMABUS),
.DSOCMBRAMBYTEWRITE (user_DSOCMBRAMBYTEWRITE),
.DSOCMBRAMEN (user_DSOCMBRAMEN),
.DSOCMBRAMWRDBUS (user_DSOCMBRAMWRDBUS),
.DSOCMBUSY (user_DSOCMBUSY),
.ISOCMBRAMEN (user_ISOCMBRAMEN),
.ISOCMBRAMEVENWRITEN (user_ISOCMBRAMEVENWRITEN),
.ISOCMBRAMODDWRITEEN (user_ISOCMBRAMODDWRITEEN),
.ISOCMBRAMRDABUS (user_ISOCMBRAMRDABUS),
.ISOCMBRAMWRBUS (user_ISOCMBRAMWRBUS),
.ISOCMBRAMWRDBUS (user_ISOCMBRAMWRDBUS),
.BRAMDSOCMCLK (user_BRAMDSOCMCLK),
.BRAMDSOCMRDDBUS (user_BRAMDSOCMRDDBUS),
.BRAMISOCMCLK (user_BRAMISOCMCLK),
.BRAMISOCMRDDBUS (user_BRAMISOCMRDDBUS),
.CPMC405CLOCK (user_CPMC405CLOCK),
.CPMC405CORECLKINACTIVE (user_CPMC405CORECLKINACTIVE),
.CPMC405CPUCUCLKEN (user_CPMC405CPUCUCLKEN),
.CPMC405JTAGCLKKEN (user_CPMC405JTAGCLKKEN),
.CPMC405TIMERCLKKEN (user_CPMC405TIMERCLKKEN),
.CPMC405TERRCLKKEN (user_CPMC405TERRCLKKEN),
.DBG245DEBUGHALT (userDBG245DEBUGHALT),
.DBG245EXTBUSHOLDACK (userDBG245EXTBUSHOLDACK),
.DBG245UNCONDDEBUGEVENT (userDBG245UNCONDDEBUGEVENT),
.DCRC245ACK (user_DCRC245ACK),
.DCRC245DBUSIN (user_DSCNTRLVALUE),
.DSCNTRLVALUE (user_DSCNTRLVALUE),
.EICC245CRITINPUTIRQ (user_EICC245CRITINPUTIRQ),
.EICC245EXTINPUTIRQ (user_EICC245EXTINPUTIRQ),
.ISARCVALUE (user_ISARCVALUE),
.ISCNTLVALUE (user_ISCNTLVALUE),
.JTGC245BDNASCANTDO (user_JTGC245BDNASCANTDO),
.JTGC245TCK (user_JTGC245TCK),
.JTGC245TDI (user_JTGC245TDI),
.JTGC245TMS (user_JTGC245TMS),
.JTGC245TRSTNEG (user_JTGC245TRSTNEG),
.MCPCPUCLKEN (user_MCPCPUCLKEN),
.MCBJTAGEN (user_MCBJTAGEN),
.MCBTIMEREN (user_MCBTIMEREN),
.MCPPCRST (user_MCPPCRST),
.PLBC405DCUADDRACK (user_PLBC405DCUADDRACK),
.PLBC405DCUBUSY (user_PLBC405DCUBUSY),
.PLBC405DCUERR (user_PLBC405DCUERR),
.PLBC405DCURDDACK (user_PLBC405DCURDDACK),
.PLBC405DCURDDDBUS (user_PLBC405DCURDDDBUS),
.PLBC405DCURDWDADDR (user_PLBC405DCURDWDADDR),
.PLBC405DCUSSIZE1 (user_PLBC405DCUSSIZE1),
.PLBC405DCUWRDACK (user_PLBC405DCUWRDACK),
.PLBC405ICUADDRACK (user_PLBC405ICUADDRACK),
.PLBC405ICUBUSY (user_PLBC405ICUBUSY),
.PLBC405ICUERR (user_PLBC405ICUERR),
.PLBC405ICURDDACK (user_PLBC405ICURDDACK),
.PLBC405ICURDDBUS (user_PLBC405ICURDDBUS),
.PLBC405ICURDWDADDR (user_PLBC405ICURDWDADDR),
.PLBC405ICUSSIZE1 (user_PLBC405ICUSSIZE1),
.PLBCLK (user_PLBCLK),
.RSTC405RESETHOOK (user_RSTC405RESETHOOK),
.RSTC405RESETCORE (user_RSTC405RESETCORE),
.RSTC405RESETSYS (user_RSTC405RESETSYS),
.TIEC405DETERMINISTICMULT (user_TIEC405DETERMINISTICMULT),
.TIEC405DISOPERANDFWD (user_TIEC405DISOPERANDFWD),
.TIEC405MMUEN (user_TIEC405MMUEN),
.TIEDSOCMDCRADDR (user_TIEDSOCMDCRADDR),
.TIEISOCMDCRADDR (user_TIEISOCMDCRADDR),
.TRCC405TRACEDISABLE (user_TRCC405TRACEDISABLE),
.TRCC405TRIGGEREVENTINE (user_TRCC405TRIGGEREVENTINE));
PULLDOWN

Resistor to GND for Input Pads

Architectures Supported

<table>
<thead>
<tr>
<th>PULLDOWN</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

PULLDOWN resistor elements are connected to input, output, or bidirectional pads to guarantee a logic Low level for nodes that might float.

Usage

For HDL, the PULLDOWN design element is instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for PULLDOWN should be placed
-- after architecture statement but before begin keyword

component PULLDOWN
    port (O : out STD_ULOGIC);
end component;

-- Component Attribute specification for PULLDOWN
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for PULLDOWN should be placed
-- in architecture after the begin keyword

PULLDOWN_INSTANCE_NAME : PULLDOWN
    port map (O => user_O);
```

Verilog Instantiation Template

```verilog
PULLDOWN instance_name (.O (user_O));
```

Commonly Used Constraints

None
PULLUP

Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs

Architectures Supported

<table>
<thead>
<tr>
<th>PULLUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>Primitive*</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
</tbody>
</table>

* Supported only on input-only pins for CoolRunner XPLA3.

The pull-up elements establish a High logic level for open-drain elements and macros (DECODE, WAND, WORAND) or 3-state nodes (TBUF) when all the drivers are off.

The buffer outputs are connected together as a wired-AND to form the output (O). When all the inputs are High, the output is off. To establish an output High level, a PULLUP resistor(s) is tied to output (O). One PULLUP resistor uses the least power, two pull-up resistors achieve the fastest Low-to-High speed.

To indicate two PULLUP resistors, append a DOUBLE parameter to the pull-up symbol attached to the output (O) node. See the appropriate CAE tool interface user guide for details.

Usage

For HDL, the PULLUP design element is instantiated rather than inferred.

VHDL Instantiation Template

```
-- Component Declaration for PULLUP should be placed
-- after architecture statement but before begin keyword

component PULLUP
  port (O : out STD_ULOGIC);
end component;

-- Component Attribute specification for PULLUP
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for PULLUP should be placed
-- in architecture after the begin keyword

PULLUP_INSTANCE_NAME : PULLUP
```

Architectures Supported

- Spartan-II, Spartan-IIE
- Spartan-3
- Virtex, Virtex-E
- Virtex-II, Virtex-II Pro, Virtex-II Pro X
- XC9500, XC9500XV, XC9500XL
- CoolRunner XPLA3
- CoolRunner-II

* Supported only on input-only pins for CoolRunner XPLA3.
port map (O => user_0);

Verilog Instantiation Template

PULLUP instance_name (.O (user_0));

Commonly Used Constraints

DOUBLE
HBLKNM
RAM16X1D

16-Deep by 1-Wide Static Dual Port Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X1D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM16X1D is a 16-word by 1-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA3 – DPRA0) and the write address (A3 – A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 4-bit write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A3-A0
data_d = word addressed by bits DPRA3-DPRA0

The SPO output reflects the data in the memory cell addressed by A3 – A0. The DPO output reflects the data in the memory cell addressed by DPRA3 – DPRA0.

Note: The write process is not affected by the address on the read address port.
Specifying Initial Contents of a RAM

You can use the INIT attribute to specify an initial value directly on the symbol if the RAM is 1 bit wide and 16, 32, 64, or 128 bits deep. The value must be a hexadecimal number, for example, INIT=ABAC. If the INIT attribute is not specified, the RAM is initialized with zero.

For Virtex, Virtex-E, Spartan-II, and Spartan-IIE, lower INIT values get mapped to the G function generator and upper INIT values get mapped to the F function generator. See the "INIT" section of the Constraints Guide for more information on the INIT attribute.

For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, wide RAMs (2, 4, and 8-bit wide single port synchronous RAMs with a WCLK) can also be initialized. These RAMs, however, require INIT_xx attributes. See “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” in the RAM16X2S section for more information on initializing Virtex-II wide RAM.

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

```vhdl
-- Component Declaration for RAM16X1D should be placed 
-- after architecture statement but before begin keyword

component RAM16X1D 
  -- synthesis translate_off
  generic (INIT : bit_vector := X"16");
  -- synthesis translate_on
  port (DPO : out STD_ULOGIC;
        SPO : out STD_ULOGIC;
        A0   : in STD_ULOGIC;
        A1   : in STD_ULOGIC;
        A2   : in STD_ULOGIC;
        A3  : in STD_ULOGIC;
        D    : in STD_ULOGIC;
        DPRA0 : in STD_ULOGIC;
        DPRA1 : in STD_ULOGIC;
        DPRA2 : in STD_ULOGIC;
        DPRA3 : in STD_ULOGIC;
        WCLK : in STD_ULOGIC;
        WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X1D 
-- should be placed after architecture declaration but 
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X1D should be placed 
-- in architecture after the begin keyword
```
Verilog Instantiation Template

RAM16X1D instance_name (.DPO (user_DPO),
 .SPO (user_SPO),
 .A0(user_A0),
 .A1(user_A1),
 .A2(user_A2),
 .A3(user_A3),
 .D (user_D),
 .DPRA0(user_DPRA0),
 .DPRA1(user_DPRA1),
 .DPRA2(user_DPRA2),
 .DPRA3(user_DPRA3),
 .WCLK(user_WCLK),
 .WE (user_WE));

defparam RAM16X1D_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM16X1D_1

16-Deep by 1-Wide Static Dual Port Synchronous RAM with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X1D_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM16X1D_1 is a 16-word by 1-bit static dual port random access memory with synchronous write capability and negative-edge clock. The device has two separate address ports: the read address (DPRA3 – DPRA0) and the write address (A3 – A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 4-bit write address. For predictable performance, write address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

You can initialize RAM16X1D_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

Data_a = word addressed by bits A3-A0
Data_d = word addressed by bits DPRA3-DPRA0

The SPO output reflects the data in the memory cell addressed by A3 – A0. The DPO output reflects the data in the memory cell addressed by DPRA3 – DPRA0.

Note: The write process is not affected by the address on the read address port.
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

```vhdl
-- Component Declaration for RAM16X1D_1 should be placed --
-- after architecture statement but before begin keyword

component RAM16X1D_1
    -- synthesis translate_off
    generic (INIT: bit_vector := X"16");
    -- synthesis translate_on
    port (DPO : out STD_ULOGIC;
         SPO : out STD_ULOGIC;
         A0   : in STD_ULOGIC;
         A1   : in STD_ULOGIC;
         A2   : in STD_ULOGIC;
         A3   : in STD_ULOGIC;
         D    : in STD_ULOGIC;
         DPRA0 : in STD_ULOGIC;
         DPRA1 : in STD_ULOGIC;
         DPRA2 : in STD_ULOGIC;
         DPRA3 : in STD_ULOGIC;
         WCLK : in STD_ULOGIC;
         WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X1D_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X1D_1 should be placed
-- in architecture after the begin keyword

RAM16X1D_1_INSTANCE_NAME : RAM16X1D_1
    -- synthesis translate_off
    generic map (INIT => hex_value)
    -- synthesis translate_on
    port map (DPO => user_DPO,
              SPO => user_SPO,
              A0 => user_A0,
              A1 => user_A1,
              A2 => user_A2,
              A3 => user_A3,
              D => user_D,
              DPRA0 => user_DPRA0,
              DPRA1 => user_DPRA1,
              DPRA2 => user_DPRA2,
              DPRA3 => user_DPRA3,
              WCLK => user_WCLK,
              WE => user_WE);
```
Verilog Instantiation Template

RAM16X1D_1 instance_name (.DPO (user_DPO),
   .SPO (user_SPO),
   .A0 (user_A0),
   .A1 (user_A1),
   .A2 (user_A2),
   .A3 (user_A3),
   .D  (user_D),
   .DPRA0 (user_DPRA0),
   .DPRA1 (user_DPRA1),
   .DPRA2 (user_DPRA2),
   .DPRA3 (user_DPRA3),
   .WCLK (user_WCLK),
   .WE  (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM16X1S

16-Deep by 1-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X1S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM16X1S is a 16-word by 1-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 4-bit address (A3 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM16X1S during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE(mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A3 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM16X1S should be placed
-- after architecture statement but before begin keyword

component RAM16X1S
  -- synthesis translate_off
generic (INIT: bit_vector := X"16");
  -- synthesis translate_on
port (O : out STD_ULOGIC;
      A0 : in STD_ULOGIC;
      A1 : in STD_ULOGIC;
      A2 : in STD_ULOGIC;
      A3 : in STD_ULOGIC;
      D : in STD_ULOGIC;
      WCLK : in STD_ULOGIC;
      WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X1S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X1S should be placed
-- in architecture after the begin keyword

RAM16X1S_INSTANCE_NAME : RAM16X1S
  -- synthesis translate_off
generic map (INIT => hex_value)
  -- synthesis translate_on
port map (O => user_O,
         A0 => user_A0,
         A1 => user_A1,
         A2 => user_A2,
         A3 => user_A3,
         D => user_D,
         WCLK => user_WCLK,
         WE => user_WE);

Verilog Instantiation Template

RAM16X1S instance_name (.O (user_O),
                        .A0 (user_A0),
                        .A1 (user_A1),
                        .A2 (user_A2),
                        .A3 (user_A3),
                        .D (user_D),
                        .WCLK (user_WCLK),
                        .WE (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints
BEL, BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, U_SET, and XBLKNM
**RAM16X1S_1**

16-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock

**Architectures Supported**

<table>
<thead>
<tr>
<th>RAM16X1S_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM16X1S_1 is a 16-word by 1-bit static random access memory with synchronous write capability and negative-edge clock. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 4-bit address (A3 – A0). For predictable performance, address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-Low WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM16X1S_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE(mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A3 – A0

**Usage**

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM16X1S_1 should be placed
-- after architecture statement but before begin keyword

component RAM16X1S_1
    -- synthesis translate_off
    generic (INIT : bit_vector := X"16");
    -- synthesis translate_on
    port (O : out STD_ULOGIC;
         A0 : in STD_ULOGIC;
         A1 : in STD_ULOGIC;
         A2 : in STD_ULOGIC;
         A3 : in STD_ULOGIC;
         D : in STD_ULOGIC;
         WCLK : in STD_ULOGIC;
         WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X1S_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X1S_1 should be placed
-- in architecture after the begin keyword

RAM16X1S_1_INSTANCE_NAME : RAM16X1S_1
    -- synthesis translate_off
    generic map (INIT =>
                  hex_value)
    -- synthesis translate_on
    port map (O => user_O,
              A0 => user_A0,
              A1 => user_A1,
              A2 => user_A2,
              A3 => user_A3,
              D => user_D,
              WCLK => user_WCLK,
              WE => user_WE);

Verilog Instantiation Template

RAM16X1S_1_instance_name (.O (user_O),
                           .A0 (user_A0),
                           .A1 (user_A1),
                           .A2 (user_A2),
                           .A3 (user_A3),
                           .D (user_D),
                           .WCLK (user_WCLK),
                           .WE (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BEL
BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
**RAM16X2D**

16-Deep by 2-Wide Static Dual Port Synchronous RAM

**Architectures Supported**

<table>
<thead>
<tr>
<th>RAM16X2D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM16X2D is a 16-word by 2-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA3 – DPRA0) and the write address (A3 – A0). These two address ports are completely asynchronous. The read address controls the location of data driven out of the output pin (DPO1 – DPO0), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D1 – D0) into the word selected by the 4-bit write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The initial contents of RAM16X2D cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

- data_a = word addressed by bits A3-A0
- data_d = word addressed by bits DPRA3-DPRA0

The SPO output reflects the data in the memory cell addressed by A3 – A0. The DPO output reflects the data in the memory cell addressed by DPRA3 – DPRA0.

**Note:** The write process is not affected by the address on the read address port.
User

For HDL, this design element is inferred. See the XST User Guide for details.
RAM16X2S

16-Deep by 2-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM16X2S is a 16-word by 2-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D1 – D0) into the word selected by the 4-bit address (A3 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O1 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

Except for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the initial contents of RAM16X2S cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, you can use the INIT_00 and INIT_01 properties to specify the initial contents of RAM16X2S as described in “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” in this section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1(read)</td>
<td>0</td>
</tr>
<tr>
<td>1(read)</td>
<td>1</td>
</tr>
<tr>
<td>1(write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A3 – A0
Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM

You can use the INIT_xx properties to specify the initial contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X wide RAM. INIT_00 initializes the RAM cells corresponding to the O0 output, INIT_01 initializes the cells corresponding to the O1 output, etc. For example, a RAM16X2S instance is initialized by INIT_00 and INIT_01 containing 4 hex characters each. A RAM16X8S instance is initialized by eight properties INIT_00 through INIT_07 containing 4 hex characters each. A RAM64x2S instance is completely initialized by two properties INIT_00 and INIT_01 containing 16 hex characters each. See the INIT_xx section of the Constraints Guide for more information on the INIT_xx attribute.

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

```vhdl
-- Component Declaration for RAM16X2S should be placed
-- after architecture statement but before begin keyword

component RAM16X2S
  -- synthesis translate_off
  generic (INIT_00 : bit_vector := X"16";
    INIT_01: bit_vector := X"16");
  -- synthesis translate_on
  port (O0 : out STD_ULOGIC;
    O1 : out STD_ULOGIC;
    A0 : in STD_ULOGIC;
    A1 : in STD_ULOGIC;
    A2 : in STD_ULOGIC;
    A3 : in STD_ULOGIC;
    D0  : in STD_ULOGIC;
    D1  : in STD_ULOGIC;
    WCLK : in STD_ULOGIC;
    WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X2S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X2S should be placed
-- in architecture after the begin keyword

RAM16X2S_INSTANCE_NAME : RAM16X2S
  -- synthesis translate_off
  generic map (INIT_00 => hex_value,
    INIT_01 => hex_value)
  -- synthesis translate_on
  port map (O0 => user_O0,
```
O1 => user_O1,
A0 => user_A0,
A1 => user_A1,
A2 => user_A2,
A3 => user_A3,
D0 => user_D0,
D1 => user_D1,
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM16X2S instance_name (.O0 (user_O0),
.O1 (user_O1),
.A0 (user_A0),
.A1 (user_A1),
.A2 (user_A2),
.A3 (user_A3)
.D0 (user_D0),
.D1 (user_D1),
.WCLK (user_WCLK),
.WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;

Commonly Used Constraints

BEL
INIT_xx
BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM16X4D

16-Deep by 4-Wide Static Dual Port Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X4D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM16X4D is a 16-word by 4-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA3 – DPRA0) and the write address (A3 – A0). These two address ports are completely asynchronous. The read address controls the location of data driven out of the output pin (DPO3 – DPO0), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D3 – D0) into the word selected by the 4-bit write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The initial contents of RAM16X4D cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A3-A0

data_d = word addressed by bits DPRA3-DPRA0

The SPO output reflects the data in the memory cell addressed by A3 – A0. The DPO output reflects the data in the memory cell addressed by DPRA3 – DPRA0.

**Note:** The write process is not affected by the address on the read address port.
Usage

For HDL, this design element must be inferred. For information on how to infer RAM, see the *XST User Guide*. 
RAM16X4S

16-Deep by 4-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X4S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM16X4S is a 16-word by 4-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D3 – D0) into the word selected by the 4-bit address (A3 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O3 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

Except for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the initial contents of RAM16X4S cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, you can use INIT_00 through INIT_03 to specify the initial contents of RAM16X4S as described in the “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” section in the RAM16X2S section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A3 – A0
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

-- Component Declaration for RAM16X4S should be placed
-- after architecture statement but before begin keyword

component RAM16X4S
  -- synthesis translate_off
  generic (INIT_00 : bit_vector := X"16";
           INIT_01 : bit_vector := X"16";
           INIT_02 : bit_vector := X"16";
           INIT_03 : bit_vector := X"16");
  -- synthesis translate_on
  port (O0  : out STD_ULOGIC;
       O1 : out STD_ULOGIC;
       O2 : out STD_ULOGIC;
       O3 : out STD_ULOGIC;
       A0   : in STD_ULOGIC;
       A1   : in STD_ULOGIC;
       A2   : in STD_ULOGIC;
       A3   : in STD_ULOGIC;
       D0  : in STD_ULOGIC;
       D1  : in STD_ULOGIC;
       D2  : in STD_ULOGIC;
       D3  : in STD_ULOGIC;
       WCLK : in STD_ULOGIC;
       WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X4S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X4S should be placed
-- in architecture after the begin keyword

RAM16X4S_INSTANCE_NAME : RAM16X4S
  -- synthesis translate_off
  generic map (INIT_00 => hex_value,
               INIT_01 => hex_value,
               INIT_02 => hex_value,
               INIT_03 => hex_value)
  -- synthesis translate_on
  port map (O0 => user_O0,
           O1 => user_O1,
           O2 => user_O2,
           O3 => user_O3,
           A0 => user_A0,
           A1 => user_A1,
           A2 => user_A2,
           A3 => user_A3,
D0 => user_D0,
D1 => user_D1,
D2 => user_D2,
D3 => user_D3,
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM16X4S instance_name (.O0 (user_O0),
 .O1 (user_O1),
 .O2 (user_O2),
 .O3 (user_O3),
.A0 (user_A0),
.A1 (user_A1),
.A2 (user_A2),
.A3 (user_A3),
.D0 (user_D0),
.D1 (user_D1),
.D2 (user_D2),
.D3 (user_D3),
.WCLK (user_WCLK),
.WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;
defparam user_instance_name.INIT_02 = hex_value;
defparam user_instance_name.INIT_03 = hex_value;

Commonly Used Constraints

BEL
INIT_xx
BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
**RAM16X8D**

16-Deep by 8-Wide Static Dual Port Synchronous RAM

**Architectures Supported**

<table>
<thead>
<tr>
<th>RAM16X8D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM16X8D is a 16-word by 8-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA3 – DPRA0) and the write address (A3 – A0). These two address ports are completely asynchronous. The read address controls the location of data driven out of the output pin (DPO7 – DPO0), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D7 – D0) into the word selected by the 4-bit write address (A3 – A0). For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The initial contents of RAM16X8D cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A3-A0  
data_d = word addressed by bits DPRA3-DPRA0

The SPO output reflects the data in the memory cell addressed by A3 – A0. The DPO output reflects the data in the memory cell addressed by DPRA3 – DPRA0.

The write process is not affected by the address on the read address port.
Usage

For HDL, this design element must be inferred. For information on how to infer RAM, see the *XST User Guide*. 
16-Deep by 8-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM16X8S</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM16X8S is a 16-word by 8-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on data inputs (D7 – D0) into the word selected by the 4-bit address (A3 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O7 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

Except for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the initial contents of RAM16X8S cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, you can use INIT_00 through INIT_07 to specify the initial contents of RAM16X8S as described in the “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” section in the RAM16X2S section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A3 – A0
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM16X8S should be placed
-- after architecture statement but before begin keyword

component RAM16X8S
-- synthesis translate_off
generic (INIT_00 : bit_vector := X"16";
  INIT_01 : bit_vector := X"16";
  INIT_02 : bit_vector := X"16";
  INIT_03 : bit_vector := X"16";
  INIT_04 : bit_vector := X"16";
  INIT_05 : bit_vector := X"16";
  INIT_06 : bit_vector := X"16";
  INIT_07 : bit_vector := X"16");
-- synthesis translate_on
port (O0 : out STD_ULOGIC;
  A0   : in STD_ULOGIC;
  A1   : in STD_ULOGIC;
  A2   : in STD_ULOGIC;
  A3   : in STD_ULOGIC;
  D    : in STD_ULOGIC;
  WCLK : in STD_ULOGIC;
  WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM16X8S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM16X8S should be placed
-- in architecture after the begin keyword

RAM16X8S_INSTANCE_NAME : RAM16X8S
-- synthesis translate_off
generic map (INIT_00 => hex_value,
  INIT_01 => hex_value,
  INIT_02 => hex_value,
  INIT_03 => hex_value,
  INIT_04 => hex_value,
  INIT_05 => hex_value,
  INIT_06 => hex_value,
  INIT_07 => hex_value)
-- synthesis translate_on
port map (O0 => user_O0,
  A0 => user_A0,
  A1 => user_A1,
  A2 => user_A2,
  A3 => user_A3,
  D => user_D,
  WCLK => user_WCLK,
  WE => user_WE);
Verilog Instantiation Template

RAM16X8S instance_name (.O0 (user_O0),
    .A0 (user_A0),
    .A1 (user_A1),
    .A2 (user_A2),
    .A3 (user_A3),
    .D (user_D),
    .WCLK (user_WCLK),
    .WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;
defparam user_instance_name.INIT_02 = hex_value;
defparam user_instance_name.INIT_03 = hex_value;
defparam user_instance_name.INIT_04 = hex_value;
defparam user_instance_name.INIT_05 = hex_value;
defparam user_instance_name.INIT_06 = hex_value;
defparam user_instance_name.INIT_07 = hex_value;

Commonly Used Constraints

BEL
INIT_xx
BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM32X1D

32-Deep by 1-Wide Static Dual Static Port Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM32X1D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X Primitive</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM32X1D is a 32-word by 1-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA4 – DPRA0) and the write address (A4 – A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 5-bit write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

You can initialize RAM32X1D during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A4-A0

data_d = word addressed by bits DPRA4-DPRA0

The SPO output reflects the data in the memory cell addressed by A4 – A0. The DPO output reflects the data in the memory cell addressed by DPRA4 – DPRA0.

Note: The write process is not affected by the address on the read address port.
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

-- Component Declaration for RAM32X1D should be placed
-- after architecture statement but before begin keyword

component RAM32X1D
-- synthesis translate_off
  generic (INIT : bit_vector := X"32");
-- synthesis translate_on
  port (DPO : out STD_ULOGIC;
       SPO : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A3 : in STD_ULOGIC;
       A4 : in STD_ULOGIC;
       D : in STD_ULOGIC;
       DPRA0 : in STD_ULOGIC;
       DPRA1 : in STD_ULOGIC;
       DPRA2 : in STD_ULOGIC;
       DPRA3 : in STD_ULOGIC;
       DPRA4 : in STD_ULOGIC;
       WCLK : in STD_ULOGIC;
       WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X1D
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X1D should be placed
-- in architecture after the begin keyword

RAM32X1D_INSTANCE_NAME : RAM32X1D
-- synthesis translate_off
  generic map (INIT => hex_value)
-- synthesis translate_on
  port map (DPO => user_DPO,
           SPO => user_SPO,
           A0 => user_A0,
           A1 => user_A1,
           A2 => user_A2,
           A3 => user_A3,
           A4 => user_A4,
           D => user_D,
           DPRA0 => user_DPRA0,
           DPRA1 => user_DPRA1,
           DPRA2 => user_DPRA2,
           DPRA3 => user_DPRA3,
           DPRA4 => user_DPRA4,
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM32X1D instance_name (.DPO (user_DPO),
  .SPO (user_SPO),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3),
  .A4 (user_A4),
  .D (user_D),
  .DPRA0 (user_DPRA0),
  .DPRA1 (user_DPRA1),
  .DPRA2 (user_DPRA2),
  .DPRA3 (user_DPRA3),
  .DPRA4 (user_DPRA4),
  .WCLK (user_WCLK),
  .WE (user_WE));

defparam RAM32X1D_instance_name.INIT = hex_value;

Commonly Used Constraints

  BLKNM
  HBLKNM
  HU_SET
  INIT
  LOC
  RLOC
  U_SET
  XBLKNM
RAM32X1D_1

32-Deep by 1-Wide Static Dual Port Synchronous RAM with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>RAM32X1D_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM32X1D_1 is a 32-word by 1-bit static dual port random access memory with synchronous write capability and a negative-edge clock. The device has two separate address ports: the read address (DPRA4 – DPRA0) and the write address (A4 – A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 5-bit write address. For predictable performance, write address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-Low WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

You can initialize RAM32X1D_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A4-A0  
data_d = word addressed by bits DPRA4-DPRA0

The SPO output reflects the data in the memory cell addressed by A4 – A0. The DPO output reflects the data in the memory cell addressed by DPRA4 – DPRA0.

Note: The write process is not affected by the address on the read address port.
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

```vhdl
-- Component Declaration for RAM32X1D_1 should be placed
-- after architecture statement but before begin keyword

component RAM32X1D_1
-- synthesis translate_off
    generic (INIT : bit_vector := X"32");
-- synthesis translate_on
    port (DPO : out STD_ULOGIC;
          SPO : out STD_ULOGIC;
          A0   : in STD_ULOGIC;
          A1   : in STD_ULOGIC;
          A2   : in STD_ULOGIC;
          A3   : in STD_ULOGIC;
          A4   : in STD_ULOGIC;
          D    : in STD_ULOGIC;
          DPRA0 : in STD_ULOGIC;
          DPRA1 : in STD_ULOGIC;
          DPRA2 : in STD_ULOGIC;
          DPRA3 : in STD_ULOGIC;
          DPRA4 : in STD_ULOGIC;
          WCLK : in STD_ULOGIC;
          WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X1D_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X1D_1 should be placed
-- in architecture after the begin keyword

RAM32X1D_1_INSTANCE_NAME : RAM32X1D_1
-- synthesis translate_off
    generic map (INIT => hex_value)
-- synthesis translate_on
    port map (DPO => user_DPO,
               SPO => user_SPO,
               A0 => user_A0,
               A1 => user_A1,
               A2 => user_A2,
               A3 => user_A3,
               A4 => user_A4,
               D => user_D,
               DPRA0 => user_DPRA0,
               DPRA1 => user_DPRA1,
               DPRA2 => user_DPRA2,
               DPRA3 => user_DPRA3,
               DPRA4 => user_DPRA4,
```
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM32X1D_1 instance_name (.DPO (user_DPO),
 .SPO (user_SPO),
 .A0 (user_A0),
 .A1 (user_A1),
 .A2 (user_A2),
 .A3 (user_A3),
 .A4 (user_A4),
 .D (user_D),
 .DPRA0 (user_DPRA0),
 .DPRA1 (user_DPRA1),
 .DPRA2 (user_DPRA2),
 .DPRA3 (user_DPRA3),
 .DPRA4 (user_DPRA4),
 .WCLK (user_WCLK),
 .WE (user_WE));

defparam RAM32X1D_1_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, U_SET, and XBLKNM
RAM32X1S

32-Deep by 1-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>RAM32X1S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM32X1S is a 32-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 5-bit address (A4 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM32X1S during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM32X1S should be placed
-- after architecture statement but before begin keyword

component RAM32X1S
  -- synthesis translate_off
generic (INIT : bit_vector := X"32");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
  A0 : in STD_ULOGIC;
  A1 : in STD_ULOGIC;
  A2 : in STD_ULOGIC;
  A4 : in STD_ULOGIC;
  D : in STD_ULOGIC;
  WCLK : in STD_ULOGIC;
  WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X1S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X1S should be placed
-- in architecture after the begin keyword

RAM32X1S_INSTANCE_NAME : RAM32X1S
  -- synthesis translate_off
generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
  A0 => user_A0,
  A1 => user_A1,
  A2 => user_A2,
  A3 => user_A3,
  A4 => user_A4,
  D => user_D,
  WCLK => user_WCLK,
  WE => user_WE);

Verilog Instantiation Template

RAM32X1S instance_name (.O (user_O),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3),
  .A4 (user_A4),
  .D (user_D),
  .WCLK (user_WCLK),
  .WE (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints
BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, U_SET, and XBLKNM
RAM32X1S_1

32-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>RAM32X1S_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM32X1S_1 is a 32-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 5-bit address (A4 – A0). For predictable performance, address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-Low WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM32X1S_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM32X1S_1 should be placed
-- after architecture statement but before begin keyword

component RAM32X1S_1
  -- synthesis translate_off
  generic (INIT : bit_vector := X"32");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A4 : in STD_ULOGIC;
       D : in STD_ULOGIC;
       WCLK : in STD_ULOGIC;
       WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X1S_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X1S_1 should be placed
-- in architecture after the begin keyword

RAM32X1S_1_INSTANCE_NAME : RAM32X1S_1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
           A0 => user_A0,
           A1 => user_A1,
           A2 => user_A2,
           A3 => user_A3,
           A4 => user_A4,
           D => user_D,
           WCLK => user_WCLK,
           WE => user_WE);
Verilog Instantiation Template

RAM32X1S_1 instance_name (.O (user_O),
    .A0 (user_A0),
    .A1 (user_A1),
    .A2 (user_A2),
    .A3 (user_A3),
    .A4 (user_A4),
    .D (user_D),
    .WCLK (user_WCLK),
    .WE (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM32X2S

32-Deep by 2-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM32X2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM32X2S is a 32-word by 2-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D1 – D0) into the word selected by the 5-bit address (A4 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O1 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

Except for Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the initial contents of RAM32X2S cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

For Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, you can use the INIT_00 and INIT_01 properties to specify the initial contents of RAM32X2S as described in “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” in the RAM16X2S section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>0 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4 – A0
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM32X2S should be placed
-- after architecture statement but before begin keyword

component RAM32X2S
-- synthesis translate_off
generic (INIT_00 : bit_vector := X"32";
        INIT_01 : bit_vector := X"32");
-- synthesis translate_on
port (O0 : out STD_ULOGIC;
     O1 : out STD_ULOGIC;
     A0 : in  STD_ULOGIC;
     A1 : in  STD_ULOGIC;
     A2 : in  STD_ULOGIC;
     A3 : in  STD_ULOGIC;
     A4 : in  STD_ULOGIC;
     D0 : in  STD_ULOGIC;
     D1 : in  STD_ULOGIC;
     WCLK : in  STD_ULOGIC;
     WE : in  STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X2S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X2S should be placed
-- in architecture after the begin keyword

RAM32X2S_INSTANCE_NAME : RAM32X2S
-- synthesis translate_off
generic map (INIT_00 => hex_value,
            INIT_01 => hex_value)
-- synthesis translate_on
port map (O0 => user_O0,
         O1 => user_O1,
         A0 => user_A0,
         A1 => user_A1,
         A2 => user_A2,
         A3 => user_A3,
         A4 => user_A4,
         D0 => user_D0,
         D1 => user_D1,
         WCLK => user_WCLK,
         WE => user_WE);
Verilog Instantiation Template

RAM32X2S instance_name (.00 (user_00),
   .01 (user_01),
   .A0 (user_A0),
   .A1 (user_A1),
   .A2 (user_A2),
   .A3 (user_A3),
   .A4 (user_A4),
   .D0 (user_D0),
   .D1 (user_D1),
   .WCLK (user_WCLK),
   .WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;

Commonly Used Constraints

INIT_xx
BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM32X4S

32-Deep by 4-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM32X4S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM32X4S is a 32-word by 4-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data inputs (D3 – D0) into the word selected by the 5-bit address (A4 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O3 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

Except for Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the initial contents of RAM32X4S cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

For Virtex-II, Virtex-II Pro, and Virtex-II Pro X, you can use the INIT_00 through INIT_03 properties to specify the initial contents of RAM32X4S as described in “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” in the RAM16X2S section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4 – A0
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM32X4S should be placed
-- after architecture statement but before begin keyword

component RAM32X4S
-- synthesis translate_off
  generic (INIT_00 : bit_vector := X"32";
   INIT_01 : bit_vector := X"32";
   INIT_02 : bit_vector := X"32";
   INIT_03 : bit_vector := X"32");
-- synthesis translate_on
  port (O0 : out STD_ULOGIC;
   O1 : out STD_ULOGIC;
   O2 : out STD_ULOGIC;
   O3 : out STD_ULOGIC;
   A0 : in STD_ULOGIC;
   A1 : in STD_ULOGIC;
   A2 : in STD_ULOGIC;
   A3 : in STD_ULOGIC;
   A4 : in STD_ULOGIC;
   D0 : in STD_ULOGIC;
   D1 : in STD_ULOGIC;
   D2 : in STD_ULOGIC;
   D3 : in STD_ULOGIC;
   WCLK : in STD_ULOGIC;
   WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X4S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X4S should be placed
-- in architecture after the begin keyword

RAM32X4S_INSTANCE_NAME : RAM32X4S
-- synthesis translate_off
  generic map (INIT_00 => hex_value,
   INIT_01 => hex_value,
   INIT_02 => hex_value,
   INIT_03 => hex_value)
-- synthesis translate_on
  port map (O0 => user_O0,
   O1 => user_O1,
   O2 => user_O2,
   O3 => user_O3,
   A0 => user_A0,
   A1 => user_A1,
   A2 => user_A2,
   A3 => user_A3,
   A4 => user_A4,
   D0 => user_D0,
   D1 => user_D1,
   D2 => user_D2,
D3 => user_D3,
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM32X4S instance_name (.O0 (user_O0),
 .O1 (user_O1),
 .O2 (user_O2),
 .O3 (user_O3),
 .A0 (user_A0),
 .A1 (user_A1),
 .A2 (user_A2),
 .A3 (user_A3),
 .A4 (user_A4),
 .D0 (user_D0),
 .D1 (user_D1),
 .D2 (user_D2),
 .D3 (user_D3),
 .WCLK (user_WCLK),
 .WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;
defparam user_instance_name.INIT_02 = hex_value;
defparam user_instance_name.INIT_03 = hex_value;

Commonly Used Constraints
INIT_xx
BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
32-Deep by 8-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM32X8S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM32X8S is a 32-word by 8-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data inputs (D7 – D0) into the word selected by the 5-bit address (A4 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O7 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

Except for Virtex-II, Virtex-II Pro, and Virtex-II Pro X, the initial contents of RAM32X8S cannot be specified directly. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

For Virtex-II, Virtex-II Pro, and Virtex-II Pro X, you can use the INIT_00 through INIT_07 properties to specify the initial contents of RAM32X8S as described in “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” in the RAM16X2S section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WE (mode)</strong></td>
<td><strong>WCLK</strong></td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A4 – A0
Ram32x8s implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM32X8S should be placed
-- after architecture statement but before begin keyword

component RAM32X8S
-- synthesis translate_off
  generic (INIT_00 : bit_vector := X"32";
  INIT_01 : bit_vector := X"32";
  INIT_02 : bit_vector := X"32";
  INIT_03 : bit_vector := X"32";
  INIT_04 : bit_vector := X"32";
  INIT_05 : bit_vector := X"32";
  INIT_06 : bit_vector := X"32";
  INIT_07 : bit_vector := X"32");
-- synthesis translate_on
  port (O0 : out STD_ULOGIC;
    A0 : in STD_ULOGIC;
    A1 : in STD_ULOGIC;
    A2 : in STD_ULOGIC;
    A3 : in STD_ULOGIC;
    A4 : in STD_ULOGIC;
    D  : in STD_ULOGIC;
    WCLK : in STD_ULOGIC;
    WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM32X8S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM32X8S should be placed
-- in architecture after the begin keyword

RAM32X8S_INSTANCE_NAME : RAM32X8S
-- synthesis translate_off
  generic map (INIT_00 => hex_value,
    INIT_01 => hex_value,
    INIT_02 => hex_value,
    INIT_03 => hex_value,
    INIT_04 => hex_value,
    INIT_05 => hex_value,
    INIT_06 => hex_value,
    INIT_07 => hex_value)
-- synthesis translate_on
  port map (O0 => user_O0,
    A0 => user_A0,
    A1 => user_A1,
    A2 => user_A2,
    A3 => user_A3,
    A4 => user_A4,
    D => user_D,
    WCLK => user_WCLK,
    WE => user_WE);
Verilog Instantiation Template

RAM32X8S instance_name (.O0 (user_O0),
   .A0 (user_A0),
   .A1 (user_A1),
   .A2 (user_A2),
   .A3 (user_A3),
   .A4 (user_A4),
   .D (user_D),
   .WCLK (user_WCLK),
   .WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;
defparam user_instance_name.INIT_02 = hex_value;
defparam user_instance_name.INIT_03 = hex_value;
defparam user_instance_name.INIT_04 = hex_value;
defparam user_instance_name.INIT_05 = hex_value;
defparam user_instance_name.INIT_06 = hex_value;
defparam user_instance_name.INIT_07 = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
INIT_xx
LOC
RLOC
U_SET
XBLKNM
RAM64X1D

64-Deep by 1-Wide Dual Port Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM64X1D</th>
<th>Spartan-II, Spartan-IIIIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

RAM64X1D is a 64-word by 1-bit static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA5 – DPRA0) and the write address (A5 – A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 6-bit (A0 - A5) write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

You can initialize RAM64X1D during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>WE (mode)</th>
<th>WCLK</th>
<th>D</th>
<th>SPO</th>
<th>DPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (read)</td>
<td>X</td>
<td>X</td>
<td>data_a</td>
<td>data_d</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
<td>X</td>
<td>data_a</td>
<td>data_d</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
<td>X</td>
<td>data_a</td>
<td>data_d</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
<td>D</td>
<td>D</td>
<td>data_d</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
<td>X</td>
<td>data_a</td>
<td>data_d</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A5-A0

data_d = word addressed by bits DPRA5-DPRA0

The SPO output reflects the data in the memory cell addressed by A5 – A0. The DPO output reflects the data in the memory cell addressed by DPRA5 – DPRA0.

Note: The write process is not affected by the address on the read address port.
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

```vhdl
-- Component Declaration for RAM64X1D should be placed
-- after architecture statement but before begin keyword

component RAM64X1D
    -- synthesis translate_off
    generic (INIT : bit_vector := X"64");
    -- synthesis translate_on
    port (DPO : out STD_ULOGIC;
          SPO : out STD_ULOGIC;
          A0 : in STD_ULOGIC;
          A1 : in STD_ULOGIC;
          A2 : in STD_ULOGIC;
          A3 : in STD_ULOGIC;
          A4 : in STD_ULOGIC;
          A5 : in STD_ULOGIC;
          D : in STD_ULOGIC;
          DPRA0 : in STD_ULOGIC;
          DPRA1 : in STD_ULOGIC;
          DPRA2 : in STD_ULOGIC;
          DPRA3 : in STD_ULOGIC;
          DPRA4 : in STD_ULOGIC;
          DPRA5 : in STD_ULOGIC;
          WCLK : in STD_ULOGIC;
          WE  : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM64X1D
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM64X1D should be placed
-- in architecture after the begin keyword

RAM64X1D_INSTANCE_NAME : RAM64X1D
    -- synthesis translate_off
    generic map(INIT => hex_value)
    -- synthesis translate_on
    port map (DPO => user_DPO,
             SPO => user_SPO,
             A0 => user_A0,
             A1 => user_A1,
             A2 => user_A2,
             A3 => user_A3,
             A4 => user_A4,
             A5 => user_A5,
             D => user_D,
             DPRA0 => user_DPRA0,
             DPRA1 => user_DPRA1,
```
DPRA2 => user_DPRA2,
DPRA3 => user_DPRA3,
DPRA4 => user_DPRA4,
DPRA5 => user_DPRA5,
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM64X1D instance_name (.DPO (user_DPO),
               .SPO (user_SPO),
               .A0 (user_A0),
               .A1 (user_A1),
               .A2 (user_A2),
               .A3 (user_A3),
               .A4 (user_A4),
               .A5 (user_A5),
               .D (user_D),
               .DPRA0 (user_DPRA0),
               .DPRA1 (user_DPRA1),
               .DPRA2 (user_DPRA2),
               .DPRA3 (user_DPRA3),
               .DPRA4 (user_DPRA4),
               .DPRA5 (user_DPRA5),
               .WCLK (user_WCLK),
               .WE (user_WE));

defparam RAM64X1D_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM64X1D_1

64-Deep by 1-Wide Dual Port Static Synchronous RAM with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>RAM64X1D_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM64X1D_1 is a 64-word by 1-bit static dual port random access memory with synchronous write capability and a negative-edge clock. The device has two separate address ports: the read address (DPRA5 – DPRA0) and the write address (A5 – A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction.

When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 6-bit (A0 - A5) write address. For predictable performance, write address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-Low WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

You can initialize RAM64X1D_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

data_a = word addressed by bits A5-A0  
data_d = word addressed by bits DPRA5-DPRA0

The SPO output reflects the data in the memory cell addressed by A5 – A0. The DPO output reflects the data in the memory cell addressed by DPRA5 – DPRA0.

**Note:** The write process is not affected by the address on the read address port.
Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template

```vhdl
-- Component Declaration for RAM64X1D_1 should be placed
-- after architecture statement but before begin keyword

component RAM64X1D_1
    -- synthesis translate_off
    generic (INIT : bit_vector := X"64");
    -- synthesis translate_on
    port (DPO : out STD_ULOGIC;
        SPO : out STD_ULOGIC;
        A0 : in STD_ULOGIC;
        A1 : in STD_ULOGIC;
        A2 : in STD_ULOGIC;
        A3 : in STD_ULOGIC;
        A4 : in STD_ULOGIC;
        A5 : in STD_ULOGIC;
        D : in STD_ULOGIC;
        DPRA0 : in STD_ULOGIC;
        DPRA1 : in STD_ULOGIC;
        DPRA2 : in STD_ULOGIC;
        DPRA3 : in STD_ULOGIC;
        DPRA4 : in STD_ULOGIC;
        DPRA5 : in STD_ULOGIC;
        WCLK : in STD_ULOGIC;
        WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM64X1D_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM64X1D_1 should be placed
-- in architecture after the begin keyword

RAM64X1D_1_INSTANCE_NAME : RAM64X1D_1
    -- synthesis translate_off
    generic map (INIT => hex_value)
    -- synthesis translate_on
    port map (DPO => user_DPO,
        SPO => user_SPO,
        A0 => user_A0,
        A1 => user_A1,
        A2 => user_A2,
        A3 => user_A3,
        A4 => user_A4,
        A5 => user_A5,
        D => user_D,
        DPRA0 => user_DPRA0,
        DPRA1 => user_DPRA1,
```
DPRA2 => user_DPRA2,
DPRA3 => user_DPRA3,
DPRA4 => user_DPRA4,
DPRA5 => user_DPRA5,
WCLK => user_WCLK,
WE => user_WE);

Verilog Instantiation Template

RAM64X1D_1 instance_name (.DPO (user_DPO),
  .SPO (user_SPO),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3),
  .A4 (user_A4),
  .A5 (user_A5),
  .D (user_D),
  .DPRA0 (user_DPRA0),
  .DPRA1 (user_DPRA1),
  .DPRA2 (user_DPRA2),
  .DPRA3 (user_DPRA3),
  .DPRA4 (user_DPRA4),
  .DPRA5 (user_DPRA5),
  .WCLK (user_WCLK),
  .WE (user_WE));

defparam RAM64X1D_1_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM64X1S

64-Deep by 1-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM64X1S</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM64X1S is a 64-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 6-bit address (A5 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM64X1S during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A5 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM64X1S should be placed
-- after architecture statement but before begin keyword

component RAM64X1S
    -- synthesis translate_off
    generic (INIT : bit_vector := X"64");
    -- synthesis translate_on
port (O : out STD_ULOGIC;
     A0 : in STD_ULOGIC;
     A1 : in STD_ULOGIC;
     A2 : in STD_ULOGIC;
     A4 : in STD_ULOGIC;
     D : in STD_ULOGIC;
     WCLK : in STD_ULOGIC;
     WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM64X1S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM64X1S should be placed
-- in architecture after the begin keyword

RAM64X1S_INSTANCE_NAME : RAM64X1S
    -- synthesis translate_off
    generic map (INIT => hex_value)
    -- synthesis translate_on
port map (O => user_O,
     A0 => user_A0,
     A1 => user_A1,
     A2 => user_A2,
     A3 => user_A3,
     A4 => user_A4,
     A5 => user_A5,
     D => user_D,
     WCLK => user_WCLK,
     WE => user_WE);
Verilog Instantiation Template

```
RAM64X1S instance_name (.O (user_O),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3),
  .A4 (user_A4),
  .A5 (user_A5),
  .D (user_D),
  .WCLK (user_WCLK),
  .WE (user_WE));
```

```
defparam user_instance_name.INIT = hex_value;
```

Commonly Used Constraints

- BLKNM
- HBLKNM
-HU_SET
-INIT
-LOC
-RLOC
-U_SET
-XBLKNM
RAM64X1S_1

64-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>RAM64X1S_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

RAM64X1S_1 is a 64-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 6-bit address (A5 – A0). For predictable performance, address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-Low WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM32X1S_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th></th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
<td>D</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
<td>D</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
<td>X</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A5 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM64X1S_1 should be placed
-- after architecture statement but before begin keyword

component RAM64X1S_1
  -- synthesis translate_off
  generic (INIT : bit_vector := X"64");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A4 : in STD_ULOGIC;
       A5 : in STD_ULOGIC;
       D : in STD_ULOGIC;
       WCLK : in STD_ULOGIC;
       WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM64X1S_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM64X1S_1 should be placed
-- in architecture after the begin keyword

RAM64X1S_1_INSTANCE_NAME : RAM64X1S_1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
           A0 => user_A0,
           A1 => user_A1,
           A2 => user_A2,
           A3 => user_A3,
           A4 => user_A4,
           A5 => user_A5,
           D => user_D,
           WCLK => user_WCLK,
           WE => user_WE);
Verilog Instantiation Template

```
RAM64X1S_1 instance_name (.O (user_0),
                              .A0 (user_A0),
                              .A1 (user_A1),
                              .A2 (user_A2),
                              .A3 (user_A3),
                              .A4 (user_A4),
                              .A5(user_A5),
                              .D (user_D),
                              .WCLK (user_WCLK),
                              .WE (user_WE));
```

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM64X2S

64-Deep by 2-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>Architectures Supported</th>
<th>RAM64X2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X Primitive</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM64X2S is a 64-word by 2-bit static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D1 – D0) into the word selected by the 6-bit address (A5 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pins (O1 – O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can use the INIT_00 and INIT_01 properties to specify the initial contents of RAM64X2S as described in “Specifying Initial Contents of a Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X Wide RAM” in the RAM16X2S section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A5 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM64X2S should be placed
-- after architecture statement but before begin keyword

component RAM64X2S
-- synthesis translate_off
    generic (INIT_00 : bit_vector := X"64";
             INIT_01 : bit_vector := X"64");
-- synthesis translate_on
    port (O0 : out STD_ULOGIC;
          O1 : out STD_ULOGIC;
          A0 : in STD_ULOGIC;
          A1 : in STD_ULOGIC;
          A2 : in STD_ULOGIC;
          A3 : in STD_ULOGIC;
          A4 : in STD_ULOGIC;
          A5 : in STD_ULOGIC;
          D0 : in STD_ULOGIC;
          D1 : in STD_ULOGIC;
          WCLK : in STD_ULOGIC;
          WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM64X2S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM64X2S should be placed
-- in architecture after the begin keyword

RAM64X2S_INSTANCE_NAME : RAM64X2S
-- synthesis translate_off
    generic map (INIT_00 => hex_value,
                  INIT_01 => hex_value)
-- synthesis translate_on
    port map (O0 => user_O0,
              O1 => user_O1,
              A0 => user_A0,
              A1 => user_A1,
              A2 => user_A2,
              A3 => user_A3,
              A4 => user_A4,
              A5 => user_A5,
              D0 => user_D0,
              D1 => user_D1,
              WCLK => user_WCLK,
              WE => user_WE);
Verilog Instantiation Template

RAM64X2S instance_name (.O0 (user_O0),
  .O1 (user_O1),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3),
  .A4 (user_A4),
  .A5 (user_A5),
  .D0 (user_D0),
  .D1 (user_D1),
  .WCLK (user_WCLK),
  .WE (user_WE));

defparam user_instance_name.INIT_00 = hex_value;
defparam user_instance_name.INIT_01 = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM128X1S

128-Deep by 1-Wide Static Synchronous RAM

Architectures Supported

<table>
<thead>
<tr>
<th>RAM128X1S</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM128X1S is a 128-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the word selected by the 7-bit address (A6 – A0). For predictable performance, address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM128X1S during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↑</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↓</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A6 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM128X1S should be placed
-- after architecture statement but before begin keyword

component RAM128X1S
  -- synthesis translate_off
  generic (INIT : bit_vector := X"128");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
  A0 : in STD_ULOGIC;
  A1 : in STD_ULOGIC;
  A2 : in STD_ULOGIC;
  A4 : in STD_ULOGIC;
  A5 : in STD_ULOGIC;
  A6 : in STD_ULOGIC;
  D : in STD_ULOGIC;
  WCLK : in STD_ULOGIC;
  WE : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM128X1S
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM128X1S should be placed
-- in architecture after the begin keyword

RAM128X1S_INSTANCE_NAME : RAM128X1S
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
  A0 => user_A0,
  A1 => user_A1,
  A2 => user_A2,
  A3 => user_A3,
  A4 => user_A4,
  A5 => user_A5,
  A6 => user_A6,
  D => user_D,
  WCLK => user_WCLK,
  WE => user_WE);
Verilog Instantiation Template

RAM128X1S instance_name (.O (user_0),
   .A0 (user_A0),
   .A1 (user_A1),
   .A2 (user_A2),
   .A3 (user_A3),
   .A4 (user_A4),
   .A5 (user_A5),
   .A6 (user_A6),
   .D (user_D),
   .WCLK (user_WCLK),
   .WE (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
RAM128X1S_1

128-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>Architectures</th>
<th>RAM128X1S_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAM128X1S_1 is a 128-word by 1-bit static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any negative transition on WCLK loads the data on the data input (D) into the word selected by the 7-bit address (A6 – A0). For predictable performance, address and data inputs must be stable before a High-to-Low WCLK transition. This RAM block assumes an active-Low WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can initialize RAM128X1S_1 during configuration using the INIT attribute. See “Specifying Initial Contents of a RAM” in the RAM16X1D section.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE (mode)</td>
<td>WCLK</td>
</tr>
<tr>
<td>0 (read)</td>
<td>X</td>
</tr>
<tr>
<td>1 (read)</td>
<td>0</td>
</tr>
<tr>
<td>1 (read)</td>
<td>1</td>
</tr>
<tr>
<td>1 (write)</td>
<td>↓</td>
</tr>
<tr>
<td>1 (read)</td>
<td>↑</td>
</tr>
</tbody>
</table>

Data = word addressed by bits A6 – A0

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template

-- Component Declaration for RAM128X1S_1 should be placed
-- after architecture statement but before begin keyword

component RAM128X1S_1
  -- synthesis translate_off
  generic (INIT : bit_vector := X"128");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A4 : in STD_ULOGIC;
       A5 : in STD_ULOGIC;
       A6 : in STD_ULOGIC;
       D : in STD_ULOGIC;
       WCLK : in STD_ULOGIC;
       WE   : in STD_ULOGIC);
end component;

-- Component Attribute specification for RAM128X1S_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for RAM128X1S_1 should be placed
-- in architecture after the begin keyword

RAM128X1S_1_INSTANCE_NAME : RAM128X1S_1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
            A0 => user_A0,
            A1 => user_A1,
            A2 => user_A2,
            A3 => user_A3,
            A4 => user_A4,
            A5 => user_A5,
            A6 => user_A6,
            D => user_D,
            WCLK => user_WCLK,
            WE => user_WE);
Verilog Instantiation Template

RAM128X1S_1 instance_name (.O (user_0),
   .A0 (user_A0),
   .A1 (user_A1),
   .A2 (user_A2),
   .A3 (user_A3),
   .A4 (user_A4),
   .A5 (user_A5),
   .A6 (user_A6),
   .D (user_D),
   .WCLK (user_WCLK),
   .WE (user_WE));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM
HBLKNM
HU_SET
INIT
LOC
RLOC
U_SET
XBLKNM
**RAMB4_Sn**

4096-Bit Single-Port Synchronous Block RAM with Port Width (n) Configured to 1, 2, 4, 8, or 16 Bits

<table>
<thead>
<tr>
<th>RAMB4_Sn</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td></td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

RAMB4_S1, RAMB4_S2, RAMB4_S4, RAMB4_S8, and RAMB4_S16 are dedicated random access memory blocks with synchronous write capability. They provide the capability for fast, discrete, large blocks of RAM in each Virtex, Virtex-E, Spartan-II, and Spartan-IIIE device. The RAMB4_Sn cell configurations are listed in the following table.

<table>
<thead>
<tr>
<th>Component</th>
<th>Depth</th>
<th>Width</th>
<th>Address Bus</th>
<th>Data Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMB4_S1</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
</tr>
<tr>
<td>RAMB4_S2</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
</tr>
<tr>
<td>RAMB4_S4</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
</tr>
<tr>
<td>RAMB4_S8</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
</tr>
<tr>
<td>RAMB4_S16</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
</tr>
</tbody>
</table>

The enable (EN) pin controls read, write, and reset. When EN is Low, no data is written and the output (DO) retains the last state. When EN is High and reset (RST) is High, DO is cleared during the Low-to-High clock (CLK) transition; if write enable (WE) is High, the memory contents reflect the data at DI. When EN is High and WE is Low, the data stored in the RAM address (ADDR) is read during the Low-to-High clock transition. When EN and WE are High, the data on the data input (DI) is loaded into the word selected by the write address (ADDR) during the Low-to-High clock transition and the data output (DO) reflects the selected (addressed) word.

The above description assumes an active High EN, WE, RST, and CLK. However, the active level can be changed by placing an inverter on the port. Any inverter placed on a RAM4 port is absorbed into the block and does not use a CLB resource.

RAMB4_Sn’s may be initialized during configuration. See “Specifying Initial Contents of a Block RAM” below.

Block RAM output registers are asynchronously cleared, output Low, when power is applied. The initial contents of the block RAM are not altered.
Virtex, Virtex-E, Spartan-II, and Spartan-IIE simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>RST</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data

Specifying Initial Contents of a Block RAM

You can use the INIT_xx attributes to specify an initial value during device configuration. The initialization of each RAMB4_Sn is set by 16 initialization attributes (INIT_00 through INIT_0F) of 64 hex values for a total of 4096 bits. See the INIT_xx section of the Constraints Guide for more information on these attributes.

If any INIT_0x attribute is not specified, it is configured as zeros. Partial initialization strings are padded with zeros to the left.

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template for RAMB4_Sn

-- Component Declaration for RAMB4_Sn
-- Should be placed after architecture statement but before begin keyword
component RAMB4_Sn
-- synthesis translate_off
generic (    INIT_00 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_01 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_02 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_03 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_04 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_05 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_06 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_08 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_09 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_0A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_0B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_0C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_0D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_0E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_0F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

-- synthesis translate_on
port (DO : out STD_LOGIC_VECTOR (0 downto 0);
      ADDR : in STD_LOGIC_VECTOR (11 downto 0);
      CLK : in STD_ULOGIC;
      DI : in STD_LOGIC_VECTOR (0 downto 0);
      EN : in STD_ULOGIC;
      RST : in STD_ULOGIC;
      WE : in STD_ULOGIC);
end component;

-- Component Attribute Specification for RAMB4_Sn
-- Should be placed after architecture declaration but before the begin keyword
-- Put attributes, if necessary

-- Component Instantiation for RAMB4_Sn
-- Should be placed in architecture after the begin keyword

RAMB4_Sn_INSTANCE_NAME : RAMB4_Sn

-- synthesis translate_off

generic map (  
  INIT_00 => hex_value,  
  INIT_01 => hex_value,  
  INIT_02 => hex_value,  
  INIT_03 => hex_value,  
  INIT_04 => hex_value,  
  INIT_05 => hex_value,  
  INIT_06 => hex_value,  
  INIT_07 => hex_value,  
  INIT_08 => hex_value,  
  INIT_09 => hex_value,  
  INIT_0A => hex_value,  
  INIT_0B => hex_value,  
  INIT_0C => hex_value,  
  INIT_0D => hex_value,  
  INIT_0E => hex_value,  
  INIT_0F => hex_value)  

-- synopsys translate_on

port map (DO => user_DO,  
  ADDR => user_ADDR,  
  CLK => user_CLK,  
  DI => user_DI,  
  EN => user_EN,  
  RST => user_RST,  
  WE => user_WE);

Verilog Instantiation Template for RAMB4_Sn

RAMB4_Sn user_instance_name (.DO (user_DO),
.ADDR (user_ADDR),
.CLK (user_CLK),
.DI (user_DI),
.EN (user_EN),
.RST (user_RST),
.WE (user_WE));

defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_0A = 256_bit_hex_value;
defparam user_instance_name.INIT_0B = 256_bit_hex_value;
defparam user_instance_name.INIT_0C = 256_bit_hex_value;
defparam user_instance_name.INIT_0D = 256_bit_hex_value;
defparam user_instance_name.INIT_0E = 256_bit_hex_value;
defparam user_instance_name.INIT_0F = 256_bit_hex_value;

Commonly Used Constraints

INIT_xx
RAMB4_Sm_Sn

4096-Bit Dual-Port Synchronous Block RAM with Port Width (m or n) Configured to 1, 2, 4, 8, or 16 Bits

Architectures Supported

<table>
<thead>
<tr>
<th>RAMB4_Sm_Sn</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>
RAMB4_Sm_Sn Representations
The RAMB4_Sm_Sn components listed in the following table are 4096-bit dual-ported dedicated random access memory blocks with synchronous write capability. Each port is independent of the other while accessing the same set of 4096 memory cells. Each port is independently configured to a specific data width.

<table>
<thead>
<tr>
<th>Component</th>
<th>Port A Depth</th>
<th>Port A Width</th>
<th>Port A ADDR</th>
<th>Port A DI</th>
<th>Port B Depth</th>
<th>Port B Width</th>
<th>Port B ADDR</th>
<th>Port B DI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMB4_S1_S1</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
</tr>
<tr>
<td>RAMB4_S1_S2</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
</tr>
<tr>
<td>RAMB4_S1_S4</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
</tr>
<tr>
<td>RAMB4_S1_S8</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
</tr>
<tr>
<td>RAMB4_S1_S16</td>
<td>4096</td>
<td>1</td>
<td>(11:0)</td>
<td>(0:0)</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
</tr>
<tr>
<td>RAMB4_S2_S2</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
</tr>
<tr>
<td>RAMB4_S2_S4</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
</tr>
<tr>
<td>RAMB4_S2_S8</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
</tr>
<tr>
<td>RAMB4_S2_S16</td>
<td>2048</td>
<td>2</td>
<td>(10:0)</td>
<td>(1:0)</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
</tr>
<tr>
<td>RAMB4_S4_S4</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
</tr>
<tr>
<td>RAMB4_S4_S8</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
</tr>
<tr>
<td>RAMB4_S4_S16</td>
<td>1024</td>
<td>4</td>
<td>(9:0)</td>
<td>(3:0)</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
</tr>
<tr>
<td>RAMB4_S8_S8</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
</tr>
<tr>
<td>RAMB4_S8_S16</td>
<td>512</td>
<td>8</td>
<td>(8:0)</td>
<td>(7:0)</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
</tr>
<tr>
<td>RAMB4_S16_S16</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
<td>256</td>
<td>16</td>
<td>(7:0)</td>
<td>(15:0)</td>
</tr>
</tbody>
</table>

ADDR=address bus for the port
DI=data input bus for the port

Each port is fully synchronous with independent clock pins. All port A input pins have setup time referenced to the CLKA pin and its data output bus DOA has a clock-to-out time referenced to the CLKA. All port B input pins have setup time referenced to the CLKB pin and its data output bus DOB has a clock-to-out time referenced to the CLKB.

The enable ENA pin controls read, write, and reset for port A. When ENA is Low, no data is written and the output (DOA) retains the last state. When ENA is High and reset (RSTA) is High, DOA is cleared during the Low-to-High clock (CLKA) transition; if write enable (WEA) is High, the memory contents reflect the data at DIA. When ENA is High and WEA is Low, the data stored in the RAM address (ADDRA) is read during the Low-to-High clock transition. When ENA and WEA are High, the data on the data input (DIA) is loaded into the word selected by the write address (ADDRA) during the Low-to-High clock transition and the data output (DOA) reflects the selected (addressed) word.

The enable ENB pin controls read, write, and reset for port B. When ENB is Low, no data is written and the output (DOB) retains the last state. When ENB is High and reset (RSTB) is High, DOB is cleared during the Low-to-High clock (CLKB) transition; if write enable (WEB) is High, the memory contents reflect the data at DIB. When ENB is High and WEB is Low, the data stored in the RAM address (ADDRB) is read during the Low-to-High clock transition. When ENB and WEB are High, the data on the data input (DIB) is loaded into the word selected by the write address (ADDRB) during the
Low-to-High clock transition and the data output (DOB) reflects the selected (addressed) word.

The above descriptions assume active High control pins (ENA, WEA, RSTA, CLKA, ENB, WEB, RSTB, and CLKB). However, the active level can be changed by placing an inverter on the port. Any inverter placed on a RAMB4 port is absorbed into the block and does not use a CLB resource.

RAMB\textsubscript{Sm}\textsubscript{Sn} s may be initialized during configuration. See the following truth table.

Block RAM output registers are asynchronously cleared, output Low, when power is applied. The initial contents of the block RAM are not altered.

Virtex, Virtex-E, Spartan-II, and Spartan-IIIE simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP\_SPARTAN2, STARTUP\_SPARTAN3, STARTUP\_VIRTEX, or STARTUP\_VIRTEX2 symbol.

Mode selection is shown in the following truth table.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN(A/B)</td>
<td>RST(A/B)</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| addr=RAM address of port A/B |
| RAM(addr)=RAM contents at address ADDRA/ADDRB |
| data=RAM input data at pins DIA/DIB |

Address Mapping

Each port accesses the same set of 4096 memory cells using an addressing scheme that is dependent on the width of the port. The physical RAM location that is addressed for a particular width is determined from the following formula.

\[
\begin{align*}
\text{Start} &= ((\text{ADDR port}+1) \times (\text{Width port})) -1 \\
\text{End} &= (\text{ADDR port} \times (\text{Width port}))
\end{align*}
\]

The following table shows address mapping for each port width.

<table>
<thead>
<tr>
<th>Port Width</th>
<th>Port Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4096</td>
</tr>
<tr>
<td>2</td>
<td>2048</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
</tr>
<tr>
<td>8</td>
<td>512</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port Width</th>
<th>Port Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00</td>
</tr>
<tr>
<td>2</td>
<td>07 06 05 04 03 02 01 00</td>
</tr>
<tr>
<td>4</td>
<td>03 02 01 00</td>
</tr>
<tr>
<td>8</td>
<td>01 00</td>
</tr>
<tr>
<td>16</td>
<td>00</td>
</tr>
</tbody>
</table>
Port A and Port B Conflict Resolution

A RAMB4_Sm_Sn component is a true dual-ported RAM in that it allows simultaneous reads of the same memory cell. When one port is performing a write to a given memory cell, the other port should not address that memory cell (for a write or a read) within the clock-to-clock setup window.

If both ports write to the same memory cell simultaneously, violating the clock-to-setup requirement, the data stored will be invalid.

If one port attempts to read from the same memory cell that the other is simultaneously writing to, violating the clock setup requirement, the write will be successful but the data read will be invalid.

Specifying Initial Contents of a Block RAM

You can use the INIT_0x attributes to specify an initial value during device configuration. The initialization of each RAMB4_Sm_Sn is set by 16 initialization attributes (INIT_00 through INIT_0F) of 64 hex values for a total of 4096 bits. See the INIT_xx section of the Constraints Guide for more information on these attributes.

If any INIT_0x attribute is not specified, it is configured as zeros. Partial initialization strings are padded with zeros to the left.

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template for RAMB4_Sm_Sn

-- Component Declaration for RAMB4_Sm_Sn
-- Should be placed after architecture statement but before begin keyword
component RAMB4_Sm_Sn
-- synthesis translate_off
  generic (  
    INIT_00 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_01 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_02 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_03 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_04 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_05 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_06 : bit_vector :=  
      X"0000000000000000000000000000000000000000000000000000000000000000";
  );
INIT_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_08 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_09 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_0A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_0B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_0C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_0D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_0E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

INIT_0F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"

-- synthesis translate_on
port (DOA : out STD_LOGIC_VECTOR (0 downto 0);
      DOB : out STD_LOGIC_VECTOR (0 downto 0);
      ADDRA : in STD_LOGIC_VECTOR (11 downto 0);
      ADDR : in STD_LOGIC_VECTOR (11 downto 0);
      CLKA : in STD_ULOGIC;
      CLKB : in STD_ULOGIC;
      DIA : in STD_LOGIC_VECTOR (0 downto 0);
      DIB : in STD_LOGIC_VECTOR (0 downto 0);
      ENA : in STD_ULOGIC;
      ENB : in STD_ULOGIC;
      RSTA : in STD_ULOGIC;
      RSTB : in STD_ULOGIC;
      WEA : in STD_ULOGIC;
      WEB : in STD_ULOGIC);
end component;
-- Component Attribute Specification for RAMB4_Sm_Sn
-- Should be placed after architecture declaration but before the begin keyword
-- Put attributes, if necessary
-- Component Instantiation for RAMB4_Sm_Sn
-- Should be placed in architecture after the begin keyword
RAMB4_Sm_Sn_INSTANCE_NAME : RAMB4_Sm_Sn

-- synthesis translate_off
generic map (  
  INIT_00 => vector_value,
  INIT_01 => vector_value,
  INIT_02 => vector_value,
  INIT_03 => vector_value,
  INIT_04 => vector_value,
  INIT_05 => vector_value,
  INIT_06 => vector_value,
  INIT_07 => vector_value,
  INIT_08 => vector_value,
  INIT_09 => vector_value,
  INIT_0A => vector_value,
  INIT_0B => vector_value,
  INIT_0C => vector_value,
  INIT_0D => vector_value,
  INIT_0E => vector_value,
  INIT_0F => vector_value)

-- synopsys translate_on
port map (DOA => user_DOA,
         DOB => user_DOB,
         ADDRA => user_ADDRA,
         ADDRAB => user_ADDRB,
         CLKA => user_CLKA,
         CLKB => user_CLKB,
         DIA => user_DIA,
         DIB => user_DIB,
         ENA => user_ENA,
         ENB => user_ENB,
         RSTA => user_RSTA,
         RSTB => user_RSTB,
         WEA => user_WEA,
WEB => user_WEB);

Verilog Instantiation Template for RAMB16_S1, S2, and S4

RAMB4_Sm_Sn user_instance_name (.DOA (user_DOA),
    .DOB (user DOB),
    .ADDRA (user_ADDRRA),
    .ADDRB (user_ADDRRB),
    .CLKA (user_CLKA),
    .CLKB (user_CLKB),
    .DIA (user_DIA),
    .DIB (user_DIB),
    .ENA (user_ENA),
    .ENB (user_ENB),
    .RSTA (user_RSTA),
    .RSTB (user_RSTB),
    .WEA (user_WEA),
    .WEB (user_WEB));

defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_0A = 256_bit_hex_value;
defparam user_instance_name.INIT_0B = 256_bit_hex_value;
defparam user_instance_name.INIT_0C = 256_bit_hex_value;
defparam user_instance_name.INIT_0D = 256_bit_hex_value;
defparam user_instance_name.INIT_0E = 256_bit_hex_value;
defparam user_instance_name.INIT_0F = 256_bit_hex_value;
Commonly Used Constraints

INIT_xx
RAMB16_Sn

16384-Bit Data Memory and 2048-Bit Parity Memory, Single-Port Synchronous Block RAM with Port Width (n) Configured to 1, 2, 4, 9, 18, or 36 Bits

Architectures Supported

<table>
<thead>
<tr>
<th>RAMB16_Sn</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-II</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

RAMB16_S1 through RAMB16_S36 Representations

RAMB16_S1, RAMB16_S2, RAMB16_S4, RAMB16_S9, RAMB16_S18, and RAMB16_S36 are dedicated random access memory blocks with synchronous write capability. The block RAM port has 16384 bits of data memory. RAMB16_S9, RAMB16_S18, and RAMB16_S36 have an additional 2048 bits of parity memory. The RAMB16_Sn cell configurations are listed in the following table.

<table>
<thead>
<tr>
<th>Component</th>
<th>Data Cells</th>
<th>Parity Cells</th>
<th>Address Bus</th>
<th>Data Bus</th>
<th>Parity Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Depth</td>
<td>Width</td>
<td>Depth</td>
<td>Width</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1</td>
<td>16384</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>(13:0)</td>
</tr>
<tr>
<td>RAMB16_S2</td>
<td>8192</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>(12:0)</td>
</tr>
<tr>
<td>Component</td>
<td>Data Cells</td>
<td>Parity Cells</td>
<td>Address Bus</td>
<td>Data Bus</td>
<td>Parity Bus</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>--------------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>Depth</td>
<td>Width</td>
<td>Depth</td>
<td>Width</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4</td>
<td>4096</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>(11:0)</td>
</tr>
<tr>
<td>RAMB16_S9</td>
<td>2048</td>
<td>8</td>
<td>2048</td>
<td>1</td>
<td>(10:0)</td>
</tr>
<tr>
<td>RAMB16_S18</td>
<td>1024</td>
<td>16</td>
<td>1024</td>
<td>2</td>
<td>(9:0)</td>
</tr>
<tr>
<td>RAMB16_S36</td>
<td>512</td>
<td>32</td>
<td>512</td>
<td>4</td>
<td>(8:0)</td>
</tr>
</tbody>
</table>
The enable (EN) pin controls read, write, and reset. When EN is Low, no data is written and the outputs (DO and DOP) retain the last state. When EN is High and reset (SSR) is High, DO and DOP are set to SRVAL during the Low-to-High clock (CLK) transition; if write enable (WE) is High, the memory contents reflect the data at DI and DIP. When SSR is Low, EN is High, and WE is Low, the data stored in the RAM address (ADDR) is read during the Low-to-High clock transition. The output value depends on the mode. By default WRITE_MODE=WRITE_FIRST, when EN and WE are High and SSR is Low, the data on the data inputs (DI and DIP) is loaded into the word selected by the write address (ADDR) during the Low-to-High clock transition. See “Write Mode Selection” for information on setting the WRITE_MODE.

The above description assumes an active High EN, WE, SSR, and CLK. However, the active level can be changed by placing an inverter on the port. Any inverter placed on a RAMB16 port is absorbed into the block and does not use a CLB resource.

<table>
<thead>
<tr>
<th>GS</th>
<th>EN</th>
<th>SSR</th>
<th>WE</th>
<th>CLK</th>
<th>ADDR</th>
<th>DI</th>
<th>DIP</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>DO: INIT</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>SRVAL</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>addr</td>
<td>data</td>
<td>pdata</td>
<td>SRVAL</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>addr</td>
<td>X</td>
<td>X</td>
<td>RAM(addr) =&gt;data</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>addr</td>
<td>data</td>
<td>pdata</td>
<td>RAM(addr) =&gt;data</td>
</tr>
</tbody>
</table>

GSR=Global Set Reset signal

INIT=Value specified by the INIT attribute for data memory. Default is all zeros.
SRVAL=Value after assertion of SSR as specified by the SRVAL attribute.

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data
pdata=RAM parity data

aWRITE_MODE=NO_CHANGE
bWRITE_MODE=READ_FIRST
cWRITE_MODE=WRITE_FIRST
Initializing Memory Contents of a Single-Port RAMB16

You can use the INIT_xx attributes to specify an initialization value for the memory contents of a RAMB16 during device configuration. The initialization of each RAMB16_Sn is set by 64 initialization attributes (INIT_00 through INIT_3F) of 64 hex values for a total of 16384 bits.

You can use the INITP_xx attributes to specify an initial value for the parity memory during device configuration or assertion. The initialization of the parity memory for ports configured for 9, 18, or 36 bits is set by 8 initialization attributes (INITP_00 through INITP_07) of 64 hex values for a total of 2048 bits.

If any INIT_xx or INITP_xx attribute is not specified, it is configured as zeros. Partial strings are padded with zeros to the left.

See the Constraints Guide for more information on these attributes.

Initializing the Output Register of a Single-Port RAMB16

In Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, each bit in the output register can be initialized at power on to either a 0 or 1. In addition, the initial state specified for power on can be different than the state that results from assertion of a set/reset. Two types of properties control initialization of the output register for a single-port RAMB16: INIT and SRVAL. The INIT attribute specifies the output register value at power on. You can use the SRVAL attribute to define the state resulting from assertion of the SSR (set/reset) input.

The INIT and SRVAL attributes specify the initialization value as a hexadecimal string. The value is dependent upon the port width. For example, for a RAMB16_S1 with port width equal to 1, the output register contains 1 bit. Therefore, the INIT or SRVAL value can only be specified as a 1 or 0. For RAMB16_S4 with port width equal to 4, the output register contains 4 bits. In this case, you can specify a hexadecimal value from 0 through F to initialize the 4 bits of the output register.

For those ports that include parity bits, the parity portion of the output register is specified in the high order bit position of the INIT or SRVAL value.

The INIT and SRVAL attributes default to zero if they are not set by the user.

See the Constraints Guide for more information on these attributes.

Write Mode Selection

The WRITE_MODE attribute controls RAMB16 memory and output contents. By default, the WRITE_MODE is set to WRITE_FIRST. This means that input is read, written to memory, and then passed to output. You can set the WRITE_MODE to READ_FIRST to read the memory contents, pass the memory contents to the outputs, and then write the input to memory. Or, you can set the WRITE_MODE to NO_CHANGE to have the input written to memory without changing the output.

Usage

For HDL, this design element can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.
VHDL Instantiation Template for RAMB16_S1, S2, and S4

-- Component Declaration for RAMB16_{S1 | S2 | S4}
-- Should be placed after architecture statement but before begin keyword
component RAMB16_{S1 | S2 | S4}
   -- synthesis translate_off
   generic (  
       INIT : bit_vector := X"0";
       INIT_00 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_01 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_02 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_03 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_04 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_05 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_06 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_07 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_08 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_09 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_0A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_0B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_0C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_0D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_0E : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_0F : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_10 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_11 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_12 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_13 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_14 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_15 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_16 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_17 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
       INIT_18 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
   );
begin
end RAMB16_{S1 | S2 | S4};
INIT_19 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1E : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1F : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_20 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_21 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_22 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_23 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_24 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_25 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_26 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_27 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_28 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_29 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2E : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2F : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_30 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_31 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_32 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_33 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_34 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_35 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
X"0000000000000000000000000000000000000000000000000000000000000000";
erves,

--- synthesis translate_on

    port (DO : out STD_LOGIC_VECTOR (0 downto 0)
        ADDR : in STD_LOGIC_VECTOR (13 downto 0);
        CLK : in STD_ULOGIC;
        DI : in STD_LOGIC_VECTOR (0 downto 0);
        EN : in STD_ULOGIC;
        SSR : in STD_ULOGIC;
        WE : in STD_ULOGIC);

end component;

-- Component Attribute Specification for RAMB16_{S1 \mid S2 \mid S4}
-- Should be placed after architecture declaration but before the begin keyword

-- Put attributes, if necessary

-- Component Instantiation for RAMB16_{S1 \mid S2 \mid S4}
-- Should be placed in architecture after the begin keyword
RAMB16_{S1 | S2 | S4}_INSTANCE_NAME : RAMB16_S1

-- synthesis translate_off

generic map (
    INIT => bit_value,
    INIT_00 => vector_value,
    INIT_01 => vector_value,
    INIT_02 => vector_value,
    INIT_03 => vector_value,
    INIT_04 => vector_value,
    INIT_05 => vector_value,
    INIT_06 => vector_value,
    INIT_07 => vector_value,
    INIT_08 => vector_value,
    INIT_09 => vector_value,
    INIT_0A => vector_value,
    INIT_0B => vector_value,
    INIT_0C => vector_value,
    INIT_0D => vector_value,
    INIT_0E => vector_value,
    INIT_0F => vector_value,
    INIT_10 => vector_value,
    INIT_11 => vector_value,
    INIT_12 => vector_value,
    INIT_13 => vector_value,
    INIT_14 => vector_value,
    INIT_15 => vector_value,
    INIT_16 => vector_value,
    INIT_17 => vector_value,
    INIT_18 => vector_value,
    INIT_19 => vector_value,
    INIT_1A => vector_value,
    INIT_1B => vector_value,
    INIT_1C => vector_value,
    INIT_1D => vector_value,
    INIT_1E => vector_value,
    INIT_1F => vector_value,
    INIT_20 => vector_value,
    INIT_21 => vector_value,
    INIT_22 => vector_value,
    INIT_23 => vector_value,
    INIT_24 => vector_value,
    INIT_25 => vector_value,
    INIT_26 => vector_value,
    INIT_27 => vector_value,
    INIT_28 => vector_value,
    INIT_29 => vector_value,
    INIT_2A => vector_value,
    INIT_2B => vector_value,
    INIT_2C => vector_value,
    INIT_2D => vector_value,
    INIT_2E => vector_value,
    INIT_2F => vector_value,
    INIT_30 => vector_value,
    INIT_31 => vector_value,
    INIT_32 => vector_value,
    INIT_33 => vector_value,
    INIT_34 => vector_value,
    INIT_35 => vector_value,
Verilog Instantiation Template for RAMB16_S1, S2, and S4

RAMB16_\{S1 | S2 | S4\} user_instance_name (.DO(user_DO),
    .ADDR (user_ADDR),
    .CLK (user_CLK),
    .DI (user_DI),
    .EN (user_EN),
    .SSR (user_SSR),
    .WE (user_WE));

defparam user_instance_name.INIT = bit_value;
defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_0A = 256_bit_hex_value;
defparam user_instance_name.INIT_0B = 256_bit_hex_value;
defparam user_instance_name.INIT_0C = 256_bit_hex_value;
defparam user_instance_name.INIT_0D = 256_bit_hex_value;
defparam user_instance_name.INIT_0E = 256_bit_hex_value;
defparam user_instance_name.INIT_0F = 256_bit_hex_value;
defparam user_instance_name.INIT_10 = 256_bit_hex_value;
defparam user_instance_name.INIT_11 = 256_bit_hex_value;
defparam user_instance_name.INIT_12 = 256_bit_hex_value;
defparam user_instance_name.INIT_13 = 256_bit_hex_value;
defparam user_instance_name.INIT_14 = 256_bit_hex_value;
defparam user_instance_name.INIT_15 = 256_bit_hex_value;
defparam user_instance_name.INIT_16 = 256_bit_hex_value;
defparam user_instance_name.INIT_17 = 256_bit_hex_value;
defparam user_instance_name.INIT_18 = 256_bit_hex_value;
defparam user_instance_name.INIT_19 = 256_bit_hex_value;
defparam user_instance_name.INIT_1A = 256_bit_hex_value;
defparam user_instance_name.INIT_1B = 256_bit_hex_value;
defparam user_instance_name.INIT_1C = 256_bit_hex_value;
defparam user_instance_name.INIT_1D = 256_bit_hex_value;
defparam user_instance_name.INIT_1E = 256_bit_hex_value;
defparam user_instance_name.INIT_1F = 256_bit_hex_value;
defparam user_instance_name.INIT_20 = 256_bit_hex_value;
defparam user_instance_name.INIT_21 = 256_bit_hex_value;
defparam user_instance_name.INIT_22 = 256_bit_hex_value;
defparam user_instance_name.INIT_23 = 256_bit_hex_value;
defparam user_instance_name.INIT_24 = 256_bit_hex_value;
defparam user_instance_name.INIT_25 = 256_bit_hex_value;
defparam user_instance_name.INIT_26 = 256_bit_hex_value;
defparam user_instance_name.INIT_27 = 256_bit_hex_value;
defparam user_instance_name.INIT_28 = 256_bit_hex_value;
defparam user_instance_name.INIT_29 = 256_bit_hex_value;
defparam user_instance_name.INIT_2A = 256_bit_hex_value;
defparam user_instance_name.INIT_2B = 256_bit_hex_value;
defparam user_instance_name.INIT_2C = 256_bit_hex_value;
defparam user_instance_name.INIT_2D = 256_bit_hex_value;
defparam user_instance_name.INIT_2E = 256_bit_hex_value;
defparam user_instance_name.INIT_2F = 256_bit_hex_value;
defparam user_instance_name.INIT_30 = 256_bit_hex_value;
defparam user_instance_name.INIT_31 = 256_bit_hex_value;
defparam user_instance_name.INIT_32 = 256_bit_hex_value;
defparam user_instance_name.INIT_33 = 256_bit_hex_value;
defparam user_instance_name.INIT_34 = 256_bit_hex_value;
defparam user_instance_name.INIT_35 = 256_bit_hex_value;
defparam user_instance_name.INIT_36 = 256_bit_hex_value;
defparam user_instance_name.INIT_37 = 256_bit_hex_value;
defparam user_instance_name.INIT_38 = 256_bit_hex_value;
defparam user_instance_name.INIT_39 = 256_bit_hex_value;
defparam user_instance_name.INIT_3A = 256_bit_hex_value;
defparam user_instance_name.INIT_3B = 256_bit_hex_value;
defparam user_instance_name.INIT_3C = 256_bit_hex_value;
defparam user_instance_name.INIT_3D = 256_bit_hex_value;
defparam user_instance_name.INIT_3E = 256_bit_hex_value;
defparam user_instance_name.INIT_3F = 256_bit_hex_value;
defparam user_instance_name.SRVAL = bit_value;
defparam user_instance_name.WRITE_MODE = write_mode;
VHDL Instantiation Template for RAMB16_S9, S18 and S36

-- Component Declaration for RAMB16_{S9 | S18 | S36}
-- Should be placed after architecture statement but before begin keyword
component RAMB16_{S9 | S18 | S36}
-- synthesis translate_off
generic (     
  INIT : bit_vector := X"0";
  INIT_00 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_01 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_02 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_03 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_04 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_05 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_06 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_07 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_08 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_09 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_0A : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_0B : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_0C : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_0D : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_0E : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_0F : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_10 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_11 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_12 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_13 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_14 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_15 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_16 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_17 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_18 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000";
);
INIT_19  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1A  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1B  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1C  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1D  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1E  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1F  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_20  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_21  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_22  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_23  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_24  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_25  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_26  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_27  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_28  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_29  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2A  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2B  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2C  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2D  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2E  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2F  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_30  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_31  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_32  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_33  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_34  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_35  :  bit_vector  :=  X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_36 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_37 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_38 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_39 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3E : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3F : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_00 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_01 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_02 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_03 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_04 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_05 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_06 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_07 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
SRVAL : bit_vector := X"0";
WRITE_MODE : string := "WRITE_FIRST";
end component;

-- Component Attribute Specification for RAMB16_{S9 | S18 | S36}
-- Should be placed after architecture declaration but before the begin keyword
-- Put attributes, if necessary
-- Component Instantiation for RAMB16_{S9 | S18 | S36}
-- Should be placed in architecture after the begin keyword

RAMB16_{S9 | S18 | S36}_INSTANCE_NAME : RAMB16_S1
-- synthesis translate_off
generic map (　
    INIT => bit_value,
    INIT_00 => vector_value,
    INIT_01 => vector_value,
    INIT_02 => vector_value,
    INIT_03 => vector_value,
    INIT_04 => vector_value,
    INIT_05 => vector_value,
    INIT_06 => vector_value,
    INIT_07 => vector_value,
    INIT_08 => vector_value,
    INIT_09 => vector_value,
    INIT_0A => vector_value,
    INIT_0B => vector_value,
    INIT_0C => vector_value,
    INIT_0D => vector_value,
    INIT_0E => vector_value,
    INIT_0F => vector_value,
    INIT_10 => vector_value,
    INIT_11 => vector_value,
    INIT_12 => vector_value,
    INIT_13 => vector_value,
    INIT_14 => vector_value,
    INIT_15 => vector_value,
    INIT_16 => vector_value,
    INIT_17 => vector_value,
    INIT_18 => vector_value,
    INIT_19 => vector_value,
    INIT_1A => vector_value,
    INIT_1B => vector_value,
    INIT_1C => vector_value,
    INIT_1D => vector_value,
    INIT_1E => vector_value,
    INIT_1F => vector_value,
    INIT_20 => vector_value,
    INIT_21 => vector_value,
    INIT_22 => vector_value,
    INIT_23 => vector_value,
    INIT_24 => vector_value,
    INIT_25 => vector_value,
    INIT_26 => vector_value,
    INIT_27 => vector_value,
    INIT_28 => vector_value,
    INIT_29 => vector_value,
    INIT_2A => vector_value,
    INIT_2B => vector_value,
    INIT_2C => vector_value,
    INIT_2D => vector_value,
    INIT_2E => vector_value,
    INIT_2F => vector_value,
    INIT_30 => vector_value,
    INIT_31 => vector_value,
INIT_32 => vector_value,
INIT_33 => vector_value,
INIT_34 => vector_value,
INIT_35 => vector_value,
INIT_36 => vector_value,
INIT_37 => vector_value,
INIT_38 => vector_value,
INIT_39 => vector_value,
INIT_3A => vector_value,
INIT_3B => vector_value,
INIT_3C => vector_value,
INIT_3D => vector_value,
INIT_3E => vector_value,
INIT_3F => vector_value,
INITP_00 => vector_value,
INITP_01 => vector_value,
INITP_02 => vector_value,
INITP_03 => vector_value,
INITP_04 => vector_value,
INITP_05 => vector_value,
INITP_06 => vector_value,
INITP_07 => vector_value,
SRVAL => bit_value,
WRITE_MODE => user_WRITE_MODE)
  -- synopsys translate_on
  port map (DO => user_DO,
    DOP => user_DOP,
    ADDR => user_ADDR,
    CLK => user_CLK,
    DI => user_DI,
    DIP => user_DIP,
    EN => user_EN,
    SSR => user_SSR,
    WE => user_WE);
Verilog Instantiation Template for RAMB16_S18 and S36

```
RAMB16_{S9 | S18 | S36} user_instance_name (.DO(user_DO),
  .DOP (user_DOP),
  .ADDR (user_ADDR),
  .CLK (user_CLK),
  .DI (user_DI),
  .DIP (user_DIP),
  .EN (user_EN),
  .SSR (user_SSR),
  .WE (user_WE));

defparam user_instance_name.INIT = bit_value;
defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_0A = 256_bit_hex_value;
defparam user_instance_name.INIT_0B = 256_bit_hex_value;
defparam user_instance_name.INIT_0C = 256_bit_hex_value;
defparam user_instance_name.INIT_0D = 256_bit_hex_value;
defparam user_instance_name.INIT_0E = 256_bit_hex_value;
defparam user_instance_name.INIT_0F = 256_bit_hex_value;
defparam user_instance_name.INIT_10 = 256_bit_hex_value;
defparam user_instance_name.INIT_11 = 256_bit_hex_value;
defparam user_instance_name.INIT_12 = 256_bit_hex_value;
defparam user_instance_name.INIT_13 = 256_bit_hex_value;
defparam user_instance_name.INIT_14 = 256_bit_hex_value;
defparam user_instance_name.INIT_15 = 256_bit_hex_value;
defparam user_instance_name.INIT_16 = 256_bit_hex_value;
defparam user_instance_name.INIT_17 = 256_bit_hex_value;
defparam user_instance_name.INIT_18 = 256_bit_hex_value;
defparam user_instance_name.INIT_19 = 256_bit_hex_value;
defparam user_instance_name.INIT_1A = 256_bit_hex_value;
defparam user_instance_name.INIT_1B = 256_bit_hex_value;
defparam user_instance_name.INIT_1C = 256_bit_hex_value;
defparam user_instance_name.INIT_1D = 256_bit_hex_value;
defparam user_instance_name.INIT_1E = 256_bit_hex_value;
defparam user_instance_name.INIT_1F = 256_bit_hex_value;
defparam user_instance_name.INIT_20 = 256_bit_hex_value;
defparam user_instance_name.INIT_21 = 256_bit_hex_value;
defparam user_instance_name.INIT_22 = 256_bit_hex_value;
defparam user_instance_name.INIT_23 = 256_bit_hex_value;
defparam user_instance_name.INIT_24 = 256_bit_hex_value;
defparam user_instance_name.INIT_25 = 256_bit_hex_value;
defparam user_instance_name.INIT_26 = 256_bit_hex_value;
defparam user_instance_name.INIT_27 = 256_bit_hex_value;
defparam user_instance_name.INIT_28 = 256_bit_hex_value;
defparam user_instance_name.INIT_29 = 256_bit_hex_value;
defparam user_instance_name.INIT_2A = 256_bit_hex_value;
defparam user_instance_name.INIT_2B = 256_bit_hex_value;
defparam user_instance_name.INIT_2C = 256_bit_hex_value;
```
defparam user_instance_name.INIT_2D = 256_bit_hex_value;
defparam user_instance_name.INIT_2E = 256_bit_hex_value;
defparam user_instance_name.INIT_2F = 256_bit_hex_value;
defparam user_instance_name.INIT_30 = 256_bit_hex_value;
defparam user_instance_name.INIT_31 = 256_bit_hex_value;
defparam user_instance_name.INIT_32 = 256_bit_hex_value;
defparam user_instance_name.INIT_33 = 256_bit_hex_value;
defparam user_instance_name.INIT_34 = 256_bit_hex_value;
defparam user_instance_name.INIT_35 = 256_bit_hex_value;
defparam user_instance_name.INIT_36 = 256_bit_hex_value;
defparam user_instance_name.INIT_37 = 256_bit_hex_value;
defparam user_instance_name.INIT_38 = 256_bit_hex_value;
defparam user_instance_name.INIT_39 = 256_bit_hex_value;
defparam user_instance_name.INIT_3A = 256_bit_hex_value;
defparam user_instance_name.INIT_3B = 256_bit_hex_value;
defparam user_instance_name.INIT_3C = 256_bit_hex_value;
defparam user_instance_name.INIT_3D = 256_bit_hex_value;
defparam user_instance_name.INIT_3E = 256_bit_hex_value;
defparam user_instance_name.INIT_3F = 256_bit_hex_value;
defparam user_instance_name.INITP_00 = 256_bit_hex_value;
defparam user_instance_name.INITP_01 = 256_bit_hex_value;
defparam user_instance_name.INITP_02 = 256_bit_hex_value;
defparam user_instance_name.INITP_03 = 256_bit_hex_value;
defparam user_instance_name.INITP_04 = 256_bit_hex_value;
defparam user_instance_name.INITP_05 = 256_bit_hex_value;
defparam user_instance_name.INITP_06 = 256_bit_hex_value;
defparam user_instance_name.INITP_07 = 256_bit_hex_value;
defparam user_instance_name.SRVAL = bit_value;
defparam user_instance_name.WRITE_MODE = write_mode;

Commonly Used Constraints

INIT
INIT_x
SRVAL
WRITE_MODE
HU_SET
INITP_x
SRVAL
WRITE_MODE
RAMB16_Sm_Sn

16384-Bit Data Memory and 2048-Bit Parity Memory, Dual-Port Synchronous Block RAM with Port Width (m or n) Configured to 1, 2, 4, 9, 18, or 36 Bits

Architectures Supported

<table>
<thead>
<tr>
<th>RAMB16_Sm_Sn</th>
<th>Spartan-II, Spartan-IIIE</th>
<th>Spartan-3</th>
<th>Virtex, Virtex-E</th>
<th>Virtex-II, Virtex-II Pro, Virtex-II Pro X</th>
<th>XC9500, XC9500XV, XC9500XL</th>
<th>CoolRunner XPLA3</th>
<th>CoolRunner-II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>Primitive</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>


RAMB16_S1_S1 through RAMB16_S1_S36 Representations
RAMB16_S2_S2 through RAMB16_S4_S36 Representations
RAMB16_Sm_Sn through RAMB16_S36_S36 Representations

The RAMB16_Sm_Sn components listed in the following table are dual-ported dedicated random access memory blocks with synchronous write capability. Each block RAM port has 16384 bits of data memory. Ports configured as 9, 18, or 36-bits wide have an additional 2048 bits of parity memory. Each port is independent of the other while accessing the same set of 16384 data memory cells. Each port is independently configured to a specific data width. The possible port and cell configurations are listed in the following table.
<table>
<thead>
<tr>
<th>Component</th>
<th>Port A</th>
<th>Port B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Cells(^d)</td>
<td>Parity Cells(^d)</td>
</tr>
<tr>
<td>RAMB16_S1_</td>
<td>16384 x 1</td>
<td>-</td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_</td>
<td>16384 x 1</td>
<td>-</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_</td>
<td>16384 x 1</td>
<td>-</td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_</td>
<td>16384 x 1</td>
<td>-</td>
</tr>
<tr>
<td>S9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_</td>
<td>16384 x 1</td>
<td>-</td>
</tr>
<tr>
<td>S18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_</td>
<td>16384 x 1</td>
<td>-</td>
</tr>
<tr>
<td>S36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_</td>
<td>8192 x 2</td>
<td>-</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_</td>
<td>8192 x 2</td>
<td>-</td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_</td>
<td>8192 x 2</td>
<td>-</td>
</tr>
<tr>
<td>S9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_</td>
<td>8192 x 2</td>
<td>-</td>
</tr>
<tr>
<td>S18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_</td>
<td>8192 x 2</td>
<td>-</td>
</tr>
<tr>
<td>S36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_</td>
<td>4096 x 4</td>
<td>-</td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_</td>
<td>4096 x 4</td>
<td>-</td>
</tr>
<tr>
<td>S9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_</td>
<td>4096 x 4</td>
<td>-</td>
</tr>
<tr>
<td>S18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_</td>
<td>4096 x 4</td>
<td>-</td>
</tr>
<tr>
<td>S36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S9_</td>
<td>2048 x 8</td>
<td>2048 x 1</td>
</tr>
<tr>
<td>S9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S9_</td>
<td>2048 x 8</td>
<td>2048 x 1</td>
</tr>
<tr>
<td>S18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S9_</td>
<td>2048 x 8</td>
<td>2048 x 1</td>
</tr>
<tr>
<td>S36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S18_</td>
<td>1024 x 16</td>
<td>1024 x 2</td>
</tr>
<tr>
<td>_S18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S18_</td>
<td>1024 x 16</td>
<td>1024 x 2</td>
</tr>
<tr>
<td>_S36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAMB16_S36_</td>
<td>512 x 32</td>
<td>512 x 4</td>
</tr>
<tr>
<td>_S36</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^d\) Depth x Width
Each port is fully synchronous with independent clock pins. All port A input pins have setup time referenced to the CLKA pin and its data output bus DOA has a clock-to-out time referenced to the CLKA. All port B input pins have setup time referenced to the CLKB pin and its data output bus DOB has a clock-to-out time referenced to the CLKB.

The enable ENA pin controls read, write, and reset for port A. When ENA is Low, no data is written and the outputs (DOA and DOPA) retain the last state. When ENA is High and reset (SSRA) is High, DOA and DOPA are set to SRVAL_A during the Low-to-High clock (CLKA) transition; if write enable (WEA) is High, the memory contents reflect the data at DIA and DIPA. When ENA is High and WEA is Low, the data stored in the RAM address (ADDRA) is read during the Low-to-High clock transition. By default, WRITE_MODE_A=WRITE_FIRST, when ENA and WEA are High, the data on the data inputs (DIA and DIPA) is loaded into the word selected by the write address (ADDRA) during the Low-to-High clock transition and the data outputs (DOA and DOPA) reflect the selected (addressed) word.

The enable ENB pin controls read, write, and reset for port B. When ENB is Low, no data is written and the outputs (DOB and DOPB) retain the last state. When ENB is High and reset (SSRB) is High, DOB and DOPB are set to SRVAL_B during the Low-to-High clock (CLKB) transition; if write enable (WEB) is High, the memory contents reflect the data at DIB and DIPB. When ENB is High and WEB is Low, the data stored in the RAM address (ADDRB) is read during the Low-to-High clock transition. By default, WRITE_MODE_B=WRITE_FIRST, when ENB and WEB are High, the data on the data inputs (DIB and DIB) are loaded into the word selected by the write address (ADDRB) during the Low-to-High clock transition and the data outputs (DOB and DOPB) reflect the selected (addressed) word.

The above descriptions assume active High control pins (ENA, WEA, SSRA, CLKA, ENB, WEB, SSRB, and CLKB). However, the active level can be changed by placing an inverter on the port. Any inverter placed on a RAMB16 port is absorbed into the block and does not use a CLB resource.

### Port A Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>RAM Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DOA</td>
<td>DOPA</td>
</tr>
<tr>
<td>GS R</td>
<td>ENA</td>
<td>SSR A WE A</td>
</tr>
<tr>
<td>1 X X X X X X X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 X X X X X X No Chg No Chg No Chg No Chg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 0 ↑ X X X SRVAL_A SRVAL_A No Chg No Chg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 1 ↑ addr data pdata SRVAL_A SRVAL_A RAM(addr) =&gt;pdata RAM(addr) =&gt;pdata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 ↑ addr X X RAM(addr) RAM(addr) No Chg No Chg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Port A Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS R</td>
<td>ENA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>GS R</td>
<td>ENB</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

GSR=Global Set Reset

INIT_A=Value specified by the INIT_A attribute for output register. Default is all zeros.
SRVAL_A=register value

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data
pdata=RAM parity data

1WRITE_MODE_A=NO_CHANGE
2WRITE_MODE_A=READ_FIRST
3WRITE_MODE_A=WRITE_FIRST

### Port B Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS R</td>
<td>ENB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

GSR=Global Set Reset

INIT_B=Value specified by the INIT_B attribute for output registers. Default is all zeros.
SRVAL_B=register value

addr=RAM address
RAM(addr)=RAM contents at address ADDR
data=RAM input data
pdata=RAM parity data

1WRITE_MODE_B=NO_CHANGE
2WRITE_MODE_B=READ_FIRST
3WRITE_MODE_B=WRITE_FIRST
Address Mapping

Each port accesses the same set of 18432 memory cells using an addressing scheme that is dependent on the width of the port. For all port widths, 16384 memory cells are available for data as shown in the “Port Address Mapping for Data” table. For 9-, 18-, and 36-bit wide ports, 2408 parity memory cells are also available as shown in “Port Address Mapping for Parity” table. The physical RAM location that is addressed for a particular width is determined from the following formula.

\[
\text{Start} = (\text{ADDR port} + 1) \times \text{Width port} \quad -1 \\
\text{End} = \text{ADDR port} \times \text{Width port}
\]

The following tables shows address mapping for each port width.

### Port Address Mapping for Data

<table>
<thead>
<tr>
<th>Data Width</th>
<th>Port Data Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16384</td>
</tr>
<tr>
<td>2</td>
<td>8192</td>
</tr>
<tr>
<td>4</td>
<td>4096</td>
</tr>
<tr>
<td>8</td>
<td>2048</td>
</tr>
<tr>
<td>16</td>
<td>1024</td>
</tr>
<tr>
<td>32</td>
<td>512</td>
</tr>
</tbody>
</table>

### Port Address Mapping for Parity

<table>
<thead>
<tr>
<th>Parity Width</th>
<th>Port Parity Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2048</td>
</tr>
<tr>
<td>2</td>
<td>1024</td>
</tr>
<tr>
<td>4</td>
<td>512</td>
</tr>
</tbody>
</table>

### Initializing Memory Contents of a Dual-Port RAMB16

You can use the INIT_xxx attributes to specify an initialization value for the memory contents of a RAMB16 during device configuration. The initialization of each RAMB16_Sm_Sn is set by 64 initialization attributes (INIT_00 through INIT_3F) of 64 hex values for a total of 16384 bits.

You can use the INITP_xxx attributes to specify an initial value for the parity memory during device configuration or assertion. The initialization of the parity memory for ports configured for 9, 18, or 36 bits is set by 8 initialization attributes (INITP_00 through INITP_07) of 64 hex values for a total of 2048 bits.

If any INIT_xxx or INITP_xxx attribute is not specified, it is configured as zeros. Partial strings are padded with zeros to the left.

See the Constraints Guide for more information on these attributes.
Initializing the Output Register of a Dual-Port RAMB16

In Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, each bit in an output register can be initialized at power on (when GSR is high) to either a 0 or 1. In addition, the initial state specified for power on can be different than the state that results from assertion of a set/reset. Four properties control initialization of the output register for a dual-port RAMB16: INIT_A, INIT_B, SRVAL_A, and SRVAL_B. The INIT_A attribute specifies the output register value at power on for port A and the INIT_B attribute specifies the value for port B. You can use the SRVAL_A attribute to define the state resulting from assertion of the SSR (set/reset) input on port A. You can use the SRVAL_B attribute to define the state resulting from assertion of the SSR input on port B.

The INIT_A, INIT_B, SRVAL_A, and SRVAL_B attributes specify the initialization value as a hexadecimal string. The value is dependent upon the port width. For example, for a RAMB16_S1_S4 with port A width equal to 1 and port B width equal to 4, the port A output register contains 1 bit and the port B output register contains 4 bits. Therefore, the INIT_A or SRVAL_A value can only be specified as a 1 or 0. For port B, the output register contains 4 bits. In this case, you can use INIT_B or SRVAL_B to specify a hexadecimal value from 0 through F to initialize the 4 bits of the output register.

For those ports that include parity bits, the parity portion of the output register is specified in the high order bit position of the INIT_A, INIT_B, SRVAL_A, or SRVAL_B value.

The INIT and SRVAL attributes default to zero if they are not set by the user. See the Constraints Guide for more information on these attributes.

Write Mode Selection

The WRITE_MODE_A attribute controls the memory and output contents of port A for a dual-port RAMB16. The WRITE_MODE_B attribute does the same for port B. By default, both WRITE_MODE_A and WRITE_MODE_B are set to WRITE_FIRST. This means that input is read, written to memory, and then passed to output. You can set the write mode for port A and/or port B to READ_FIRST to read the memory contents, pass the memory contents to the outputs, and then write the input to memory. Or, you can set the write mode to NO_CHANGE to have the input written to memory without changing the output. The “Port A and Port B Conflict Resolution” section describes how read/write conflicts are resolved when both port A and port B are attempting to read/write to the same memory cells.

Port A and Port B Conflict Resolution

Spartan-3, Virtex-II, Virtex-II Pro, and Virtex-II Pro X block SelectRAM is True Dual-Port RAM that allows both ports to simultaneously access the same memory cell. When one port writes to a given memory cell, the other port must not address that memory cell (for a write or a read) within the clock-to-clock setup window. For a list of specifics of conflict resolution for port and memory cell write operations that have either a clock common to both ports or synchronous clocks on each port, see Virtex-II Handbook, Chapter 2, Design Considerations, Using BlockSelectRAM Memory, Conflict Resolution.

The following tables summarize the collision detection behavior of the dual-port RAMB16 based on the WRITE_MODE_A and WRITE_MODE_B settings.
### WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=NO_CHANGE

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data RAM</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>X</td>
<td>No Chg</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>X</td>
<td>No Chg</td>
<td>X</td>
<td>No Chg</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>No Chg</td>
<td>No Chg</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### WRITE_MODE_A=READ_FIRST and WRITE_MODE_B=READ_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data RAM</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>DIA</td>
<td>X</td>
<td>DIPA</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>X</td>
<td>X</td>
<td>DIB</td>
<td>DIPB</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### WRITE_MODE_A= WRITE_FIRST and WRITE_MODE_B=WRITE_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data RAM</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>X</td>
<td>No Chg</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>No Chg</td>
<td>X</td>
<td>X</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
</tbody>
</table>

### WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=READ_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data RAM</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>X</td>
<td>No Chg</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>No Chg</td>
<td>X</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
</tbody>
</table>

### WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=WRITE_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data RAM</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>DIA</td>
<td>X</td>
<td>DIPA</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>X</td>
<td>X</td>
<td>DIB</td>
<td>DIPB</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### WRITE_MODE_A=WRITE_FIRST and WRITE_MODE_B=WRITE_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data RAM</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>X</td>
<td>No Chg</td>
<td>X</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>No Chg</td>
<td>X</td>
<td>X</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
</tbody>
</table>
WRITE_MODE_A=NO_CHANGE and WRITE_MODE_B=WRITE_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data Ram</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>X</td>
<td>DIB</td>
<td>X</td>
<td>DIPB</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>No Chg</td>
<td>X</td>
<td>No Chg</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

WRITE_MODE_A=READ_FIRST and WRITE_MODE_B=WRITE_FIRST

<table>
<thead>
<tr>
<th>WEA</th>
<th>WEB</th>
<th>CLKA</th>
<th>CLKB</th>
<th>DIA</th>
<th>DIB</th>
<th>DIPA</th>
<th>DIPB</th>
<th>DOA</th>
<th>DOB</th>
<th>DOPA</th>
<th>DOPB</th>
<th>Data Ram</th>
<th>Parity Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>X</td>
<td>DIB</td>
<td>X</td>
<td>DIPB</td>
<td>DIB</td>
<td>DIPB</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>↑</td>
<td>DIA</td>
<td>DIB</td>
<td>DIPA</td>
<td>DIPB</td>
<td>X</td>
<td>DIB</td>
<td>X</td>
<td>DIPB</td>
<td>DIA</td>
<td>DIPA</td>
</tr>
</tbody>
</table>

Usage

For HDL, these design elements can be inferred or instantiated. The instantiation code is shown below. For information on how to infer RAM, see the XST User Guide.

VHDL Instantiation Template for RAMB16_S1_S1, RAMB16_S1_S2, RAMB16_S1_S4, RAMB16_S2_S2, RAMB16_S2_S4, and RAMB16_S4_S4

-- Component Declaration for RAMB16_S1_{S1 | S2 | S4}, RAMB16_S2_{S2 | S4}, and RAMB16_S4_S4 should be placed after architecture statement but before begin keyword

-- For the following component declaration, enter RAMB16_S1_{S1 | S2 | S4},
-- RAMB16_S2_{S2 | S4}, or RAMB16_S4_S4

colorial component RAMB16_Sm_Sn

-- synthesis translate_off
generic (  
  INIT_00 : bit_vector :=  
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_01 : bit_vector :=  
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_02 : bit_vector :=  
    X"0000000000000000000000000000000000000000000000000000000000000000";
  INIT_03 : bit_vector :=  
    X"0000000000000000000000000000000000000000000000000000000000000000";
);
INIT_04 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_05 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_06 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_08 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_09 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_10 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_11 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_12 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_13 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_14 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_15 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_16 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_17 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_18 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_19 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_20 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_21 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_22 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_23 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_24 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_25 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_26 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_27 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_28 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_29 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_30 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_31 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_32 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_33 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_34 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_35 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_36 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_37 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_38 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_39 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_A : bit_vector := X"0";
INIT_B : bit_vector := X"0";
INITP_00 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_01 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_02 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"
;
INITP_03 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"
;
INITP_04 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"
;
INITP_05 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"
;
INITP_06 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"
;
INITP_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"
;
SRVAL_A : bit_vector := X"0";
SRVAL_B : bit_vector := X"0";
WRITE_MODE_A : string := "WRITE_FIRST";
WRITE_MODE_B : string := "WRITE_FIRST";

);  
-- synthesis translate_on

port (DOA : out STD_LOGIC_VECTOR (n downto 0);
      DOB : out STD_LOGIC_VECTOR (n downto 0);
      ADDRA : in STD_LOGIC_VECTOR (n downto 0);
      ADDR_B : in STD_LOGIC_VECTOR (n downto 0);
      CLKA : in STD_ULOGIC;
      CLKB : in STD_ULOGIC;
      DIA : in STD_LOGIC_VECTOR (n downto 0);
      DIB : in STD_LOGIC_VECTOR (n downto 0);
      ENA : in STD_ULOGIC;
      ENB : in STD_ULOGIC;
      SSRA : in STD_ULOGIC;
      SSRB : in STD_ULOGIC;
      WEA : in STD_ULOGIC;
      WEB : in STD_ULOGIC);

end component;
-- Component Attribute Specification for design element
-- should be placed after architecture declaration
-- but before the begin keyword

-- Put attributes, if necessary

-- Component Instantiation for design element
-- Should be placed in architecture after the begin keyword

RAMB16_Sm_Sn INSTANCE_NAME : RAMB16_Sm_Sn
-- synthesis translate_off
generic map (    INIT_00 => vector_value,
    INIT_01 => vector_value,
    INIT_02 => vector_value,
    INIT_03 => vector_value,
    INIT_04 => vector_value,
    INIT_05 => vector_value,
    INIT_06 => vector_value,
    INIT_07 => vector_value,
    INIT_08 => vector_value,
    INIT_09 => vector_value,
    INIT_0A => vector_value,
    INIT_0B => vector_value,
    INIT_0C => vector_value,
    INIT_0D => vector_value,
    INIT_0E => vector_value,
    INIT_0F => vector_value,
    INIT_10 => vector_value,
    INIT_11 => vector_value,
    INIT_12 => vector_value,
    INIT_13 => vector_value,
    INIT_14 => vector_value,
    INIT_15 => vector_value,
INIT_16 => vector_value,
INIT_17 => vector_value,
INIT_18 => vector_value,
INIT_19 => vector_value,
INIT_1A => vector_value,
INIT_1B => vector_value,
INIT_1C => vector_value,
INIT_1D => vector_value,
INIT_1E => vector_value,
INIT_1F => vector_value,
INIT_20 => vector_value,
INIT_21 => vector_value,
INIT_22 => vector_value,
INIT_23 => vector_value,
INIT_24 => vector_value,
INIT_25 => vector_value,
INIT_26 => vector_value,
INIT_27 => vector_value,
INIT_28 => vector_value,
INIT_29 => vector_value,
INIT_2A => vector_value,
INIT_2B => vector_value,
INIT_2C => vector_value,
INIT_2D => vector_value,
INIT_2E => vector_value,
INIT_2F => vector_value,
INIT_30 => vector_value,
INIT_31 => vector_value,
INIT_32 => vector_value,
INIT_33 => vector_value,
INIT_34 => vector_value,
INIT_35 => vector_value,
INIT_36 => vector_value,
INIT_37 => vector_value,
INIT_38 => vector_value,
INIT_39 => vector_value,
INIT_3A => vector_value,
INIT_3B => vector_value,
INIT_3C => vector_value,
INIT_3D => vector_value,
INIT_3E => vector_value,
INIT_3F => vector_value,
INIT_A => bit_value,
INIT_B => bit_value,
INITP_00 => vector_value,
INITP_01 => vector_value,
INITP_02 => vector_value,
INITP_03 => vector_value,
INITP_04 => vector_value,
INITP_05 => vector_value,
INITP_06 => vector_value,
INITP_07 => vector_value,
SRVAL_A => bit_value,
SRVAL_B => bit_value,
WRITE_MODE_A => string_value,
WRITE_MODE_B => string_value)

-- synopsys translate_on
port map (DOA => user_DOA,
      DOB => user_DOB,
      ADDRA => user_ADDRA,
      ADDR_B => user_ADDRB,
      CLKA => user_CLKA,
      CLKB => user_CLKB,
      DIA => user_DIA,
      DIB => user_DIB,
      ENA => user_ENA,
      ENB => user_ENB,
      SSRA => user_SSRA,
      SSRB => user_SSRB,
      WEA => user_WEA,
WEB => user_WEB);

Verilog Instantiation Template for RAMB16_S1_S1, RAMB16_S1_S2, RAMB16_S1_S4, RAMB16_S2_S2, RAMB16_S2_S4, and RAMB16_S4_S4

RAMB16_Sm_Sn user_instance_name (.DOA (user_DOA),
  .DOB (user_DOB),
  .ADDRA (userADDRA),
  .ADDRB (userADDRB),
  .CLKA (user_CLKA),
  .CLKB (user_CLKB),
  .DIA (user_DIA),
  .DIB (user_DIB),
  .ENA (user_ENA),
  .ENB (user_ENB),
  .SSRA (user_SSRA),
  .SSRB (user_SSRB),
  .WEA (user_WEA),
  .WEB (user_WEB));

defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_0A = 256_bit_hex_value;
defparam user_instance_name.INIT_0B = 256_bit_hex_value;
defparam user_instance_name.INIT_0C = 256_bit_hex_value;
defparam user_instance_name.INIT_0D = 256_bit_hex_value;
defparam user_instance_name.INIT_0E = 256_bit_hex_value;
defparam user_instance_name.INIT_0F = 256_bit_hex_value;
defparam user_instance_name.INIT_10 = 256_bit_hex_value;
defparam user_instance_name.INIT_11 = 256_bit_hex_value;
defparam user_instance_name.INIT_12 = 256_bit_hex_value;
defparam user_instance_name.INIT_13 = 256_bit_hex_value;
defparam user_instance_name.INIT_14 = 256_bit_hex_value;
defparam user_instance_name.INIT_15 = 256_bit_hex_value;
defparam user_instance_name.INIT_16 = 256_bit_hex_value;
defparam user_instance_name.INIT_17 = 256_bit_hex_value;
defparam user_instance_name.INIT_18 = 256_bit_hex_value;
defparam user_instance_name.INIT_19 = 256_bit_hex_value;
defparam user_instance_name.INIT_1A = 256_bit_hex_value;
defparam user_instance_name.INIT_1B = 256_bit_hex_value;
defparam user_instance_name.INIT_1C = 256_bit_hex_value;
defparam user_instance_name.INIT_1D = 256_bit_hex_value;
defparam user_instance_name.INIT_1E = 256_bit_hex_value;
defparam user_instance_name.INIT_1F = 256_bit_hex_value;
defparam user_instance_name.INIT_20 = 256_bit_hex_value;
defparam user_instance_name.INIT_21 = 256_bit_hex_value;
defparam user_instance_name.INIT_22 = 256_bit_hex_value;
defparam user_instance_name.INIT_23 = 256_bit_hex_value;
defparam user_instance_name.INIT_24 = 256_bit_hex_value;
defparam user_instance_name.INIT_25 = 256_bit_hex_value;
defparam user_instance_name.INIT_26 = 256_bit_hex_value;
defparam user_instance_name.INIT_27 = 256_bit_hex_value;
defparam user_instance_name.INIT_28 = 256_bit_hex_value;
defparam user_instance_name.INIT_29 = 256_bit_hex_value;
defparam user_instance_name.INIT_2A = 256_bit_hex_value;
defparam user_instance_name.INIT_2B = 256_bit_hex_value;
defparam user_instance_name.INIT_2C = 256_bit_hex_value;
defparam user_instance_name.INIT_2D = 256_bit_hex_value;
defparam user_instance_name.INIT_2E = 256_bit_hex_value;
defparam user_instance_name.INIT_2F = 256_bit_hex_value;
defparam user_instance_name.INIT_30 = 256_bit_hex_value;
defparam user_instance_name.INIT_31 = 256_bit_hex_value;
defparam user_instance_name.INIT_32 = 256_bit_hex_value;
defparam user_instance_name.INIT_33 = 256_bit_hex_value;
defparam user_instance_name.INIT_34 = 256_bit_hex_value;
defparam user_instance_name.INIT_35 = 256_bit_hex_value;
defparam user_instance_name.INIT_36 = 256_bit_hex_value;
defparam user_instance_name.INIT_37 = 256_bit_hex_value;
defparam user_instance_name.INIT_38 = 256_bit_hex_value;
defparam user_instance_name.INIT_39 = 256_bit_hex_value;
defparam user_instance_name.INIT_3A = 256_bit_hex_value;
defparam user_instance_name.INIT_3B = 256_bit_hex_value;
defparam user_instance_name.INIT_3C = 256_bit_hex_value;
defparam user_instance_name.INIT_3D = 256_bit_hex_value;
defparam user_instance_name.INIT_3E = 256_bit_hex_value;
defparam user_instance_name.INIT_3F = 256_bit_hex_value;
defparam user_instance_name.INIT_A = bit_value;
defparam user_instance_name.INIT_B = bit_value;
defparam user_instance_name.INITP_00 = 256_bit_hex_value;
defparam user_instance_name.INITP_01 = 256_bit_hex_value;
defparam user_instance_name.INITP_02 = 256_bit_hex_value;
defparam user_instance_name.INITP_03 = 256_bit_hex_value;
defparam user_instance_name.INITP_04 = 256_bit_hex_value;
defparam user_instance_name.INITP_05 = 256_bit_hex_value;
defparam user_instance_name.INITP_06 = 256_bit_hex_value;
defparam user_instance_name.INITP_07 = 256_bit_hex_value;
defparam user_instance_name.SRVAL_A = bit_value;
defparam user_instance_name.SRVAL_B = bit_value;
defparam user_instance_name.WRITE_MODE_A = string_value;
defparam user_instance_name.WRITE_MODE_B = string_value;

VHDL Instantiation Template for RAMB16_S1_S9, RAMB16_S1_S18, RAMB16_S1_S36, RAMB16_S2_S9, RAMB16_S2_S18, RAMB16_S2_S36, RAMB16_S4_S9, RAMB16_S4_S18, and RAMB16_S4_S36

-- Component Declaration for these design elements
-- should be placed after architecture statement but before begin keyword

-- For the following component declaration, enter RAMB16_S1_{S9 | S18 | S36},
component RAMB16_Sm_Sn
  -- synthesis translate_off
  generic (
    INIT_00 : bit_vector :=
      X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_01 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_02 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_03 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_04 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_05 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_06 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_07 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_08 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_09 : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_0A : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_0B : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_0C : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_0D : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_0E : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
  INIT_0F : bit_vector :=
    X"0000000000000000000000000000000000000000000000000000000000000000"
  );
INIT_12 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_13 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_14 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_15 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_16 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_17 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_18 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_19 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_20 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_21 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_22 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_23 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_24 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_25 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_26 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_27 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_28 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_29 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2E : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_2F : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_30 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_31 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_32 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_33 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_34 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_35 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_36 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_37 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_38 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_39 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
```
INIT_3C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_A : bit_vector := X"0";
INIT_B : bit_vector := X"0";
INITP_00 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_01 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_02 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_03 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_04 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_05 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_06 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
SRVAL_A : bit_vector := X"0";
SRVAL_B : bit_vector := X"0";
WRITE_MODE_A : string := "WRITE_FIRST";
WRITE_MODE_B : string := "WRITE_FIRST";
);
-- synthesis translate_on

port (DOA : out STD_LOGIC_VECTOR (n downto 0);
     DOB : out STD_LOGIC_VECTOR (n downto 0);
     DOPB : out STD_LOGIC_VECTOR (n downto 0);
     ADDRA : in STD_LOGIC_VECTOR (n downto 0);
     ADDRB : in STD_LOGIC_VECTOR (n downto 0);
     CLKA : in STD_ULOGIC;
```
CLKB : in STD_ULOGIC;
DIA : in STD_LOGIC_VECTOR (n downto 0);
DIB : in STD_LOGIC_VECTOR (n downto 0);
DIPB : in STD_LOGIC_VECTOR (n downto 0);
ENA : in STD_ULOGIC;
ENB : in STD_ULOGIC;
SSRA : in STD_ULOGIC;
SSRB : in STD_ULOGIC;
WEA : in STD_ULOGIC;
WEB : in STD_ULOGIC);

end component;

-- Component Attribute Specification for design element
-- should be placed after architecture declaration
-- but before the begin keyword

-- Put attributes, if necessary
-- Component Instantiation for design element
-- Should be placed in architecture after the begin keyword

RAMB16_Sm_Sn INSTANCE_NAME : RAMB16_Sm_Sn
-- synthesis translate_off
generic map ( 
  INIT_00 => vector_value,
  INIT_01 => vector_value,
  INIT_02 => vector_value,
  INIT_03 => vector_value,
  INIT_04 => vector_value,
  INIT_05 => vector_value,
  INIT_06 => vector_value,
  INIT_07 => vector_value,
  INIT_08 => vector_value,
  INIT_09 => vector_value,
  INIT_0A => vector_value,
INIT_0B => vector_value,
INIT_0C => vector_value,
INIT_0D => vector_value,
INIT_0E => vector_value,
INIT_0F => vector_value,
INIT_10 => vector_value,
INIT_11 => vector_value,
INIT_12 => vector_value,
INIT_13 => vector_value,
INIT_14 => vector_value,
INIT_15 => vector_value,
INIT_16 => vector_value,
INIT_17 => vector_value,
INIT_18 => vector_value,
INIT_19 => vector_value,
INIT_1A => vector_value,
INIT_1B => vector_value,
INIT_1C => vector_value,
INIT_1D => vector_value,
INIT_1E => vector_value,
INIT_1F => vector_value,
INIT_20 => vector_value,
INIT_21 => vector_value,
INIT_22 => vector_value,
INIT_23 => vector_value,
INIT_24 => vector_value,
INIT_25 => vector_value,
INIT_26 => vector_value,
INIT_27 => vector_value,
INIT_28 => vector_value,
INIT_29 => vector_value,
INIT_2A => vector_value,
INIT_2B => vector_value,
INIT_2C => vector_value,
INIT_2D => vector_value,
INIT_2E => vector_value,
INIT_2F => vector_value,
INIT_30 => vector_value,
INIT_31 => vector_value,
INIT_32 => vector_value,
INIT_33 => vector_value,
INIT_34 => vector_value,
INIT_35 => vector_value,
INIT_36 => vector_value,
INIT_37 => vector_value,
INIT_38 => vector_value,
INIT_39 => vector_value,
INIT_3A => vector_value,
INIT_3B => vector_value,
INIT_3C => vector_value,
INIT_3D => vector_value,
INIT_3E => vector_value,
INIT_3F => vector_value,
INIT_A => bit_value,
INIT_B => bit_value,
INITP_00 => vector_value,
INITP_01 => vector_value,
INITP_02 => vector_value,
INITP_03 => vector_value,
INITP_04 => vector_value,
INITP_05 => vector_value,
INITP_06 => vector_value,
INITP_07 => vector_value,
SRVAL_A => bit_value,
SRVAL_B => bit_value,
WRITE_MODE_A => string_value,
WRITE_MODE_B => string_value)

-- synopsys translate_on

port map (DOA => user_DOA,
         DOB => user_DOB,
Verilog Instantiation Template for RAMB16_S1_S9, RAMB16_S1_S18, RAMB16_S1_S36, RAMB16_S2_S9, RAMB16_S2_S18, RAMB16_S2_S36, RAMB16_S4_S9, RAMB16_S4_S18, and RAMB16_S4_S36

RAMB16_Sm_Sn user_instance_name (.DOA (user_DOA),
   .DOB (user_DOB),
   .DOPB (user_DOPB),
   .ADDRA (user_ADDRA),
   .ADDRB (user_ADDRB),
   .CLKA (user_CLKA),
   .CLKB (user_CLKB),
   .DIA (user_DIA),
   .DIB (user_DIB),
   .DIPB (user_DIPB),
   .ENA (user_ENA),
   .ENB (user_ENB),
   .SSRA (user_SSRA),
   .SSRB (user_SSRB),
   .WEA (user_WEA),
   .WEB (user_WEB));
defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_A0 = 256_bit_hex_value;
defparam user_instance_name.INIT_A1 = 256_bit_hex_value;
defparam user_instance_name.INIT_A2 = 256_bit_hex_value;
defparam user_instance_name.INIT_A3 = 256_bit_hex_value;
defparam user_instance_name.INIT_A4 = 256_bit_hex_value;
defparam user_instance_name.INIT_A5 = 256_bit_hex_value;
defparam user_instance_name.INIT_A6 = 256_bit_hex_value;
defparam user_instance_name.INIT_A7 = 256_bit_hex_value;
defparam user_instance_name.INIT_A8 = 256_bit_hex_value;
defparam user_instance_name.INIT_A9 = 256_bit_hex_value;
defparam user_instance_name.INIT_AF = 256_bit_hex_value;
defparam user_instance_name.INIT_B0 = 256_bit_hex_value;
defparam user_instance_name.INIT_B1 = 256_bit_hex_value;
defparam user_instance_name.INIT_B2 = 256_bit_hex_value;
defparam user_instance_name.INIT_B3 = 256_bit_hex_value;
defparam user_instance_name.INIT_B4 = 256_bit_hex_value;
defparam user_instance_name.INIT_B5 = 256_bit_hex_value;
defparam user_instance_name.INIT_B6 = 256_bit_hex_value;
defparam user_instance_name.INIT_B7 = 256_bit_hex_value;
defparam user_instance_name.INIT_B8 = 256_bit_hex_value;
defparam user_instance_name.INIT_B9 = 256_bit_hex_value;
defparam user_instance_name.INIT_BF = 256_bit_hex_value;
defparam user_instance_name.INIT_C0 = 256_bit_hex_value;
defparam user_instance_name.INIT_C1 = 256_bit_hex_value;
defparam user_instance_name.INIT_C2 = 256_bit_hex_value;
defparam user_instance_name.INIT_C3 = 256_bit_hex_value;
defparam user_instance_name.INIT_C4 = 256_bit_hex_value;
defparam user_instance_name.INIT_C5 = 256_bit_hex_value;
defparam user_instance_name.INIT_C6 = 256_bit_hex_value;
defparam user_instance_name.INIT_C7 = 256_bit_hex_value;
defparam user_instance_name.INIT_C8 = 256_bit_hex_value;
defparam user_instance_name.INIT_C9 = 256_bit_hex_value;
defparam user_instance_name.INIT_CE = 256_bit_hex_value;
defparam user_instance_name.INIT_D0 = 256_bit_hex_value;
defparam user_instance_name.INIT_D1 = 256_bit_hex_value;
defparam user_instance_name.INIT_D2 = 256_bit_hex_value;
defparam user_instance_name.INIT_D3 = 256_bit_hex_value;
defparam user_instance_name.INIT_D4 = 256_bit_hex_value;
defparam user_instance_name.INIT_D5 = 256_bit_hex_value;
defparam user_instance_name.INIT_D6 = 256_bit_hex_value;
defparam user_instance_name.INIT_D7 = 256_bit_hex_value;
defparam user_instance_name.INIT_D8 = 256_bit_hex_value;
defparam user_instance_name.INIT_D9 = 256_bit_hex_value;
defparam user_instance_name.INIT_DF = 256_bit_hex_value;
defparam user_instance_name.INIT_E0 = 256_bit_hex_value;
defparam user_instance_name.INIT_E1 = 256_bit_hex_value;
defparam user_instance_name.INIT_E2 = 256_bit_hex_value;
defparam user_instance_name.INIT_E3 = 256_bit_hex_value;
defparam user_instance_name.INIT_E4 = 256_bit_hex_value;
defparam user_instance_name.INIT_E5 = 256_bit_hex_value;
defparam user_instance_name.INIT_E6 = 256_bit_hex_value;
defparam user_instance_name.INIT_E7 = 256_bit_hex_value;
defparam user_instance_name.INIT_E8 = 256_bit_hex_value;
defparam user_instance_name.INIT_E9 = 256_bit_hex_value;
defparam user_instance_name.INIT_EC = 256_bit_hex_value;
defparam user_instance_name.INIT_F0 = 256_bit_hex_value;
defparam user_instance_name.INIT_F1 = 256_bit_hex_value;
defparam user_instance_name.INIT_F2 = 256_bit_hex_value;
defparam user_instance_name.INIT_F3 = 256_bit_hex_value;
defparam user_instance_name.INIT_F4 = 256_bit_hex_value;
defparam user_instance_name.INIT_F5 = 256_bit_hex_value;
defparam user_instance_name.INIT_F6 = 256_bit_hex_value;
defparam user_instance_name.INIT_F7 = 256_bit_hex_value;
defparam user_instance_name.INIT_F8 = 256_bit_hex_value;
defparam user_instance_name.INIT_F9 = 256_bit_hex_value;
defparam user_instance_name.INIT_FF = 256_bit_hex_value;
defparam user_instance_name.INIT_23 = 256_bit_hex_value;
defparam user_instance_name.INIT_24 = 256_bit_hex_value;
defparam user_instance_name.INIT_25 = 256_bit_hex_value;
defparam user_instance_name.INIT_26 = 256_bit_hex_value;
defparam user_instance_name.INIT_27 = 256_bit_hex_value;
defparam user_instance_name.INIT_28 = 256_bit_hex_value;
defparam user_instance_name.INIT_29 = 256_bit_hex_value;
defparam user_instance_name.INIT_2A = 256_bit_hex_value;
defparam user_instance_name.INIT_2B = 256_bit_hex_value;
defparam user_instance_name.INIT_2C = 256_bit_hex_value;
defparam user_instance_name.INIT_2D = 256_bit_hex_value;
defparam user_instance_name.INIT_2E = 256_bit_hex_value;
defparam user_instance_name.INIT_2F = 256_bit_hex_value;
defparam user_instance_name.INIT_30 = 256_bit_hex_value;
defparam user_instance_name.INIT_31 = 256_bit_hex_value;
defparam user_instance_name.INIT_32 = 256_bit_hex_value;
defparam user_instance_name.INIT_33 = 256_bit_hex_value;
defparam user_instance_name.INIT_34 = 256_bit_hex_value;
defparam user_instance_name.INIT_35 = 256_bit_hex_value;
defparam user_instance_name.INIT_36 = 256_bit_hex_value;
defparam user_instance_name.INIT_37 = 256_bit_hex_value;
defparam user_instance_name.INIT_38 = 256_bit_hex_value;
defparam user_instance_name.INIT_39 = 256_bit_hex_value;
defparam user_instance_name.INIT_3A = 256_bit_hex_value;
defparam user_instance_name.INIT_3B = 256_bit_hex_value;
defparam user_instance_name.INIT_3C = 256_bit_hex_value;
defparam user_instance_name.INIT_3D = 256_bit_hex_value;
defparam user_instance_name.INIT_3E = 256_bit_hex_value;
defparam user_instance_name.INIT_3F = 256_bit_hex_value;
defparam user_instance_name.INIT_A = bit_value;
defparam user_instance_name.INIT_B = bit_value;
defparam user_instance_name.INITP_00 = 256_bit_hex_value;
defparam user_instance_name.INITP_01 = 256_bit_hex_value;
defparam user_instance_name.INITP_02 = 256_bit_hex_value;
defparam user_instance_name.INITP_03 = 256_bit_hex_value;
defparam user_instance_name.INITP_04 = 256_bit_hex_value;
defparam user_instance_name.INITP_05 = 256_bit_hex_value;
defparam user_instance_name.INITP_06 = 256_bit_hex_value;
defparam user_instance_name.INITP_07 = 256_bit_hex_value;
defparam user_instance_name.SRVAL_A = bit_value;
defparam user_instance_name.SRVAL_B = bit_value;
defparam user_instance_name.WRITE_MODE_A = string_value;
defparam user_instance_name.WRITE_MODE_B = string_value;

VHDL Instantiation Template RAMB16_S9_S9, RAMB16_S9_S18, RAMB16_S9_S36, RAMB16_S18_S18, RAMB16_S18_S36, and RAMB16_S36_S36
-- Component Declaration for these design elements
-- should be placed after architecture statement but before begin keyword

-- For the following component declaration, enter RAMB16_S9_[S9 | S18 | S36],
-- RAMB16_S18_[S18 | S36], or RAMB16_S36_S36

component RAMB16_Sm_Sn
-- synthesis translate_off
generic (  
    INIT_00 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_01 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_02 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_03 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_04 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_05 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_06 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_07 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
    INIT_08 : bit_vector :=  
        X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_09 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0E : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_0F : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_10 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_11 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_12 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_13 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_14 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_15 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_16 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_17 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_18 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_19 : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1A : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1B : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1C : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1D : bit_vector := X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_1E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_1F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_20 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_21 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_22 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_23 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_24 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_25 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_26 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_27 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_28 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_29 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_2F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_30 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_31 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_32 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_33 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_34 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_35 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_36 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_37 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_38 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_39 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_3F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INIT_A : bit_vector := X"0";

INIT_B : bit_vector := X"0";

INITP_00 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_01 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_02 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_03 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_04 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_05 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";

INITP_06 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INITP_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000";
SRVAL_A : bit_vector := X"0";
SRVAL_B : bit_vector := X"0";
WRITE_MODE_A : string := "WRITE_FIRST";
WRITE_MODE_B : string := "WRITE_FIRST";
);
-- synthesis translate_on

port (DOA : out STD_LOGIC_VECTOR (n downto 0);
DOB : out STD_LOGIC_VECTOR (n downto 0);
DOPA : out STD_LOGIC_VECTOR (n downto 0);
DOPB : out STD_LOGIC_VECTOR (n downto 0);
ADDRA : in STD_LOGIC_VECTOR (n downto 0);
ADDRB : in STD_LOGIC_VECTOR (n downto 0);
CLKA : in STD_ULOGIC;
CLKB : in STD_ULOGIC;
DIA : in STD_LOGIC_VECTOR (n downto 0);
DIB : in STD_LOGIC_VECTOR (n downto 0);
DIPA : in STD_LOGIC_VECTOR (n downto 0);
DIPB : in STD_LOGIC_VECTOR (n downto 0);
ENA: in STD_ULOGIC;
ENB : in STD_ULOGIC;
SSRA : in STD_ULOGIC;
SSRB : in STD_ULOGIC;
WEA : in STD_ULOGIC;
WEB : in STD_ULOGIC);

end component;

-- Component Attribute Specification for design element
-- should be placed after architecture declaration
-- but before the begin keyword

-- Put attributes, if necessary
-- Component Instantiation for design element
-- Should be placed in architecture after the begin keyword

RAMB16_Sm_Sn INSTANCE_NAME : RAMB16_Sm_Sn
-- synthesis translate_off
generic map (
  INIT_00 => vector_value,
  INIT_01 => vector_value,
  INIT_02 => vector_value,
  INIT_03 => vector_value,
  INIT_04 => vector_value,
  INIT_05 => vector_value,
  INIT_06 => vector_value,
  INIT_07 => vector_value,
  INIT_08 => vector_value,
  INIT_09 => vector_value,
  INIT_0A => vector_value,
  INIT_0B => vector_value,
  INIT_0C => vector_value,
  INIT_0D => vector_value,
  INIT_0E => vector_value,
  INIT_0F => vector_value,
  INIT_10 => vector_value,
  INIT_11 => vector_value,
  INIT_12 => vector_value,
  INIT_13 => vector_value,
  INIT_14 => vector_value,
  INIT_15 => vector_value,
  INIT_16 => vector_value,
  INIT_17 => vector_value,
  INIT_18 => vector_value,
  INIT_19 => vector_value,
  INIT_1A => vector_value,
  INIT_1B => vector_value,
INIT_1C => vector_value,
INIT_1D => vector_value,
INIT_1E => vector_value,
INIT_1F => vector_value,
INIT_20 => vector_value,
INIT_21 => vector_value,
INIT_22 => vector_value,
INIT_23 => vector_value,
INIT_24 => vector_value,
INIT_25 => vector_value,
INIT_26 => vector_value,
INIT_27 => vector_value,
INIT_28 => vector_value,
INIT_29 => vector_value,
INIT_2A => vector_value,
INIT_2B => vector_value,
INIT_2C => vector_value,
INIT_2D => vector_value,
INIT_2E => vector_value,
INIT_2F => vector_value,
INIT_30 => vector_value,
INIT_31 => vector_value,
INIT_32 => vector_value,
INIT_33 => vector_value,
INIT_34 => vector_value,
INIT_35 => vector_value,
INIT_36 => vector_value,
INIT_37 => vector_value,
INIT_38 => vector_value,
INIT_39 => vector_value,
INIT_3A => vector_value,
INIT_3B => vector_value,
INIT_3C => vector_value,
INIT_3D => vector_value,
INIT_3E => vector_value,
INIT_3F => vector_value,
INIT_A => bit_value,
INIT_B => bit_value,
INITP_00 => vector_value,
INITP_01 => vector_value,
INITP_02 => vector_value,
INITP_03 => vector_value,
INITP_04 => vector_value,
INITP_05 => vector_value,
INITP_06 => vector_value,
INITP_07 => vector_value,
SRVAL_A => bit_value,
SRVAL_B => bit_value,
WRITE_MODE_A => string_value,
WRITE_MODE_B => string_value)
-- synopsys translate_on
port map (DOA => user_DOA,
     DOB => user DOB,
     DOPA => user_DOPA,
     DOPB => user_DOPB,
     ADDRA => user_ADDRA,
     ADDRB => user_ADDRB,
     CLKA => user_CLKA,
     CLKB => user_CLKB,
     DIA => user_DIA,
     DIB => user_DIB,
     DIPA => user_DIPA,
     DIPB => user_DIPB,
     ENA => user_ENA,
     ENB => user_ENB,
     SSRA => user_SSRA,
     SSRB => user_SSRB,
     WEA => user_WEA,
     WEB => user_WEB);
Verilog Instantiation Template for RAMB16_S9_S9, RAMB16_S9_S18, RAMB16_S9_S36, RAMB16_S18_S18, RAMB16_S18_S36, and RAMB16_S36_S36

Ramb16_Sm_Sn user_instance_name (.DOA (user.DOA),
  .DOB (user.DOB),
  .DOPA (user.DOPA),
  .DOPB (user.DOPB),
  .ADDRA (user.ADDRA),
  .ADDRB (user.ADDRB),
  .CLKA (user.CLKA),
  .CLKB (user.CLKB),
  .DIA (user.DIA),
  .DIB (user.DIB),
  .DIPA (user.DIPA),
  .DIPB (user.DIPB),
  .ENA (user.ENA),
  .ENB (user.ENB),
  .SSRA (user.SSRA),
  .SSRB (user.SSRB),
  .WEA (user.WEA),
  .WEB (user.WEB));

defparam user_instance_name.INIT_00 = 256_bit_hex_value;
defparam user_instance_name.INIT_01 = 256_bit_hex_value;
defparam user_instance_name.INIT_02 = 256_bit_hex_value;
defparam user_instance_name.INIT_03 = 256_bit_hex_value;
defparam user_instance_name.INIT_04 = 256_bit_hex_value;
defparam user_instance_name.INIT_05 = 256_bit_hex_value;
defparam user_instance_name.INIT_06 = 256_bit_hex_value;
defparam user_instance_name.INIT_07 = 256_bit_hex_value;
defparam user_instance_name.INIT_08 = 256_bit_hex_value;
defparam user_instance_name.INIT_09 = 256_bit_hex_value;
defparam user_instance_name.INIT_0A = 256_bit_hex_value;
defparam user_instance_name.INIT_0B = 256_bit_hex_value;
defparam user_instance_name.INIT_0C = 256_bit_hex_value;
defparam user_instance_name.INIT_0D = 256_bit_hex_value;
defparam user_instance_name.INIT_0E = 256_bit_hex_value;
defparam user_instance_name.INIT_0F = 256_bit_hex_value;
defparam user_instance_name.INIT_10 = 256_bit_hex_value;
defparam user_instance_name.INIT_11 = 256_bit_hex_value;
defparam user_instance_name.INIT_12 = 256_bit_hex_value;
defparam user_instance_name.INIT_13 = 256_bit_hex_value;
defparam user_instance_name.INIT_14 = 256_bit_hex_value;
defparam user_instance_name.INIT_15 = 256_bit_hex_value;
defparam user_instance_name.INIT_16 = 256_bit_hex_value;
defparam user_instance_name.INIT_17 = 256_bit_hex_value;
defparam user_instance_name.INIT_18 = 256_bit_hex_value;
defparam user_instance_name.INIT_19 = 256_bit_hex_value;
defparam user_instance_name.INIT_1A = 256_bit_hex_value;
defparam user_instance_name.INIT_1B = 256_bit_hex_value;
defparam user_instance_name.INIT_1C = 256_bit_hex_value;
defparam user_instance_name.INIT_1D = 256_bit_hex_value;
defparam user_instance_name.INIT_1E = 256_bit_hex_value;
defparam user_instance_name.INIT_1F = 256_bit_hex_value;
defparam user_instance_name.INIT_20 = 256_bit_hex_value;
defparam user_instance_name.INIT_21 = 256_bit_hex_value;
defparam user_instance_name.INIT_22 = 256_bit_hex_value;
defparam user_instance_name.INIT_23 = 256_bit_hex_value;
defparam user_instance_name.INIT_24 = 256_bit_hex_value;
defparam user_instance_name.INIT_25 = 256_bit_hex_value;
defparam user_instance_name.INIT_26 = 256_bit_hex_value;
defparam user_instance_name.INIT_27 = 256_bit_hex_value;
defparam user_instance_name.INIT_28 = 256_bit_hex_value;
defparam user_instance_name.INIT_29 = 256_bit_hex_value;
defparam user_instance_name.INIT_2A = 256_bit_hex_value;
defparam user_instance_name.INIT_2B = 256_bit_hex_value;
defparam user_instance_name.INIT_2C = 256_bit_hex_value;
defparam user_instance_name.INIT_2D = 256_bit_hex_value;
defparam user_instance_name.INIT_2E = 256_bit_hex_value;
defparam user_instance_name.INIT_2F = 256_bit_hex_value;
defparam user_instance_name.INIT_30 = 256_bit_hex_value;
defparam user_instance_name.INIT_31 = 256_bit_hex_value;
defparam user_instance_name.INIT_32 = 256_bit_hex_value;
defparam user_instance_name.INIT_33 = 256_bit_hex_value;
defparam user_instance_name.INIT_34 = 256_bit_hex_value;
defparam user_instance_name.INIT_35 = 256_bit_hex_value;
defparam user_instance_name.INIT_36 = 256_bit_hex_value;
defparam user_instance_name.INIT_37 = 256_bit_hex_value;
defparam user_instance_name.INIT_38 = 256_bit_hex_value;
defparam user_instance_name.INIT_39 = 256_bit_hex_value;
defparam user_instance_name.INIT_3A = 256_bit_hex_value;
defparam user_instance_name.INIT_3B = 256_bit_hex_value;
defparam user_instance_name.INIT_3C = 256_bit_hex_value;
defparam user_instance_name.INIT_3D = 256_bit_hex_value;
defparam user_instance_name.INIT_3E = 256_bit_hex_value;
defparam user_instance_name.INIT_3F = 256_bit_hex_value;
defparam user_instance_name.INIT_A = bit_value;
defparam user_instance_name.INIT_B = bit_value;
defparam user_instance_name.INITP_00 = 256_bit_hex_value;
defparam user_instance_name.INITP_01 = 256_bit_hex_value;
defparam user_instance_name.INITP_02 = 256_bit_hex_value;
defparam user_instance_name.INITP_03 = 256_bit_hex_value;
defparam user_instance_name.INITP_04 = 256_bit_hex_value;
defparam user_instance_name.INITP_05 = 256_bit_hex_value;
defparam user_instance_name.INITP_06 = 256_bit_hex_value;
defparam user_instance_name.INITP_07 = 256_bit_hex_value;
defparam user_instance_name.SRVAL_A = bit_value;
defparam user_instance_name.SRVAL_B = bit_value;
defparam user_instance_name.WRITE_MODE_A = string_value;
defparam user_instance_name.WRITE_MODE_B = string_value;

Commonly Used Constraints
INIT, INIT_xx, INIT_A, INIT_B, INTP_xx, SRVAL_A, SRVAL_B, WRITE_MODE_A, WRITE_MODE_B
ROC

Reset On Configuration

Architectures Supported

<table>
<thead>
<tr>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

The ROC is a component used for VHDL simulation of FPGA designs. This component should not be used for Verilog or schematic entry. The ROC’s function is to mimic the function of the internal reset signal during the FPGA configuration process. In order to use ROC, it must be connected to the reset/preset signal for all inferred and instantiated registers in the design. During synthesis and implementation, this reset signal will use the dedicated global set/reset network and will not use local routing resources. During simulation, ROC will emit a one-shot pulse for the amount of time specified by the WIDTH generic (default is 100 ns). This one-shot pulse is intended to reset all registers so that at the beginning of operation, all registers are at a known value as would happen in the real silicon during configuration of the device.

For more information, see the Xilinx Synthesis and Verification Design Guide.

Port O will be high at simulation time 0 for the amount of time specified by the WIDTH generic attribute. After that time, it will be 0. This will not affect implementation in any way.

VHDL Instantiation Code

```vhdl
component ROC
  -- synthesis translate_off
  generic (WIDTH : Time := 100 ns);
  -- synthesis translate_on
  port (O : out std_ulogic := '1');
end component;
```

Commonly Used Constraints

For simulation, the WIDTH generic can be modified to change the amount of time the one-shot pulse is applied for.

There are no supported constraints for this component for implementation.
ROCBUF

Reset On Configuration Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>ROCBUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

The ROCBUF is a component used for VHDL simulation of FPGA designs that is similar to the ROC component except that it contains an input for controlling the global set/reset function rather than a one-shot. This component should not be used for Verilog or schematic entry. The ROCBUF’s function allows user control of the function of the global set/reset signal as done during the FPGA configuration process. In order to use the ROCBUF, the input should be connected to a top-level port in the design and the output must be connected to the reset/preset signal for all inferred and instantiated registers in the design.

During simulation, the input to the ROCBUF can be toggled by the testbench in order to activate the global set/reset signal in the design. This should be done at the beginning of the simulation as is done in the real silicon after configuration to get the design in a known state. The signal may also be pulsed during simulation to simulate a reconfiguration (ProG pin high) of the device. During synthesis and implementation, this reset signal will use the dedicated global set/reset network and will not use local routing resources. The port connected to this component will be optimized out of the design and not use any pin resources.

If you want to have the port implemented in the design, a STARTBUF_architecture should be used. In order to replace this port during back-end simulation the -gp switch should be used when invoking the netgen. If using the ISE GUI, use the "Bring Out Global Set/Reset Net as a Port" option in the Simulation Model Properties window.

For more information, see the *Xilinx Synthesis and Verification Design Guide*.

The value at port O will always be the value at port I (it is a buffer).

VHDL Instantiation Code

```vhdl
component ROCBUF
  port ( I : in  std_ulogic;
         O : out std_ulogic);
end component;
```

Commonly Used Constraints

None
ROM16X1

16-Deep by 1-Wide ROM

Architectures Supported

<table>
<thead>
<tr>
<th>ROM16X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

ROM16X1 is a 16-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 4-bit address (A3 – A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of four hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H. For example, the INIT=10A7 parameter produces the data stream:

```
0001 0000 1010 0111
```

An error occurs if the INIT=value is not specified. See the appropriate CAE tool interface user guide for details.

Usage

For HDL, the ROM16X1 design element should be instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for ROM16X1 should be placed
-- after architecture statement but before begin keyword

component ROM16X1
    -- synthesis translate_off
    generic (INIT : bit_vector := X"16");
    -- synthesis translate_on
    port (O : out STD_ULOGIC;
         A0 : in STD_ULOGIC;
         A1 : in STD_ULOGIC;
         A2 : in STD_ULOGIC;
         A3 : in STD_ULOGIC);
end component;

-- Component Attribute specification for ROM16X1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
```
-- Component Instantiation for ROM16X1 should be placed
-- in architecture after the begin keyword

ROM16X1_INSTANCE_NAME : ROM16X1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
    A0 => user_A0,
    A1 => user_A1,
    A2 => user_A2,
    A3 => user_A3);

Verilog Instantiation Template

ROM16X1 instance_name (.O (user_O),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, ROM_EXTRACT, U_SET, XBLKNM
ROM32X1

32-Deep by 1-Wide ROM

Architectures Supported

<table>
<thead>
<tr>
<th>ROM32X1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ROM32X1 is a 32-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 5-bit address (A4 – A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of eight hexadecimal digits that are written into the ROM from the most-significant digit A=1FH to the least-significant digit A=00H. For example, the INIT=10A78F39 parameter produces the data stream:

```
0001 0000 1010 0111 1000 1111 0011 1001
```

An error occurs if the INIT=value is not specified. See the appropriate CAE tool interface user guide for details.

Usage

For HDL, the ROM32X1 design element should be instantiated rather than inferred.

VHDL Instantiation Template

```
-- Component Declaration for ROM32X1 should be placed
-- after architecture statement but before begin keyword

component ROM32X1
  -- synthesis translate_off
  generic (INIT : bit_vector := X"32");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
    A0 : in STD_ULOGIC;
    A1 : in STD_ULOGIC;
    A2 : in STD_ULOGIC;
    A3 : in STD_ULOGIC;
    A4 : in STD_ULOGIC);
end component;

-- Component Attribute specification for ROM32X1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
```
-- Component Instantiation for ROM32X1 should be placed
-- in architecture after the begin keyword
ROM32X1_INSTANCE_NAME : ROM32X1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (O => user_O,
           A0 => user_A0,
           A1 => user_A1,
           A2 => user_A2,
           A3 => user_A3,
           A4 => user_A4);

Verilog Instantiation Template

ROM32X1 instance_name (.O (user_O),
                         .A0 (user_A0),
                         .A1 (user_A1),
                         .A2 (user_A2),
                         .A3 (user_A3),
                         .A4 (user_A4));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints
BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, ROM_EXTRACT, U_SET, XBLKNM
ROM64X1

64-Deep by 1-Wide ROM

Architectures Supported

<table>
<thead>
<tr>
<th>ROM64X1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ROM64X1 is a 64-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 6-bit address (A5 – A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of 16 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if the INIT=value is not specified. See the appropriate CAE tool interface user guide for details.

Usage

For HDL, the ROM64X1 design element should be instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for ROM64X1 should be placed
-- after architecture statement but before begin keyword

component ROM64X1

-- synthesis translate_off
generic (INIT : bit_vector := X"64");
-- synthesis translate_on
port (O : out STD_ULOGIC;
    A0 : in STD_ULOGIC;
    A1 : in STD_ULOGIC;
    A2 : in STD_ULOGIC;
    A3 : in STD_ULOGIC;
    A4 : in STD_ULOGIC;
    A5 : in STD_ULOGIC);
end component;

-- Component Attribute specification for ROM64X1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
```
-- Component Instantiation for ROM64X1 should be placed
-- in architecture after the begin keyword

ROM64X1_INSTANCE_NAME : ROM64X1
   -- synthesis translate_off
   generic map (INIT => hex_value)
   -- synthesis translate_on
   port map (O => user_O,
             A0 => user_A0,
             A1 => user_A1,
             A2 => user_A2,
             A3 => user_A3,
             A4 => user_A4,
             A5 => user_A5);

Verilog Instantiation Template

ROM64X1 instance_name (.O (user_O),
                      .A0 (user_A0),
                      .A1 (user_A1),
                      .A2 (user_A2),
                      .A3 (user_A3),
                      .A4 (user_A4),
                      .A5 (user_A5));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, ROM_EXTRACT, U_SET, XBLKNM
ROM128X1

128-Deep by 1-Wide ROM

Architectures Supported

<table>
<thead>
<tr>
<th>ROM128X1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

ROM128X1 is a 128-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 7-bit address (A6 – A0). The ROM is initialized to a known value during configuration with the INIT= value parameter. The value consists of 32 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if the INIT= value is not specified. See the appropriate CAE tool interface user guide for details.

Usage

For HDL, the ROM128X1 design element should be instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for ROM128X1 should be placed
-- after architecture statement but before begin keyword

component ROM128X1
  -- synthesis translate_off
  generic (INIT : bit_vector := X"128");
  -- synthesis translate_on
  port (O : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A3 : in STD_ULOGIC;
       A4 : in STD_ULOGIC;
       A5 : in STD_ULOGIC;
       A6 : in STD_ULOGIC);
end component;

-- Component Attribute specification for ROM128X1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
```
-- Component Instantiation for ROM128X1 should be placed
-- in architecture after the begin keyword
ROM128X1_INSTANCE_NAME : ROM128X1
  -- synthesis translate_off
generic map (INIT => hex_value)
  -- synthesis translate_on
port map (O => user_O,
  A0 => user_A0,
  A1 => user_A1,
  A2 => user_A2,
  A3 => user_A3,
  A4 => user_A4,
  A5 => user_A5,
  A6 => user_A6);

Verilog Instantiation Template

ROM128X1 instance_name (.O (user_O),
  .A0 (user_A0),
  .A1 (user_A1),
  .A2 (user_A2),
  .A3 (user_A3),
  .A4 (user_A4),
  .A5 (user_A5),
  .A6 (user_A6));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, ROM_EXTRACT, U_SET, XBLKNM
ROM256X1

256-Deep by 1-Wide ROM

Architectures Supported

<table>
<thead>
<tr>
<th>ROM256X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

ROM256X1 is a 256-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 8-bit address (A7–A0). The ROM is initialized to a known value during configuration with the INIT=value parameter. The value consists of 64 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if the INIT=value is not specified. See the appropriate CAE tool interface user guide for details.

Usage

For HDL, the ROM256X1 design element should be instantiated rather than inferred.

VHDL Instantiation Template

-- Component Declaration for ROM256X1 should be placed
-- after architecture statement but before begin keyword

component ROM256X1
   -- synthesis translate_off
   generic (INIT : bit_vector := X"256");
   -- synthesis translate_on
   port (O : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A3 : in STD_ULOGIC;
       A4 : in STD_ULOGIC;
       A5 : in STD_ULOGIC;
       A6 : in STD_ULOGIC;
       A7 : in STD_ULOGIC);
end component;

-- Component Attribute specification for ROM256X1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here
-- Component Instantiation for ROM256X1 should be placed
-- in architecture after the begin keyword

ROM256X1_INSTANCE_NAME : ROM256X1
-- synthesis translate_off
generic map (INIT => hex_value)
-- synthesis translate_on
port map (O=> user_O,
A0 => user_A0,
A1 => user_A1,
A2 => user_A2,
A3 => user_A3,
A4 => user_A4,
A5 => user_A5,
A6 => user_A6,
A7 => user_A7);

Verilog Instantiation Template

ROM256X1 instance_name (.O (user_O),
.A0 (user_A0),
.A1 (user_A1),
.A2 (user_A2),
.A3 (user_A3),
.A4 (user_A4),
.A5(user_A5),
.A6(user_A6),
.A7(user_A7));

defparam user_instance_name.INIT = hex_value;

Commonly Used Constraints

BLKNM, HBLKNM, HU_SET, INIT, LOC, RLOC, ROM_EXTRACT, U_SET, XBLKNM
Sum of Products

Architectures Supported

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SOP4B4

SOP4B3

SOP4B2B

SOP4B2A

SOP4B1

SOP4B3

SOP3B3

SOP3B2B

SOP3B2A

SOP3B1B

SOP3B1A

SOP3

SOP4


SOP Gate Representations

Sum Of Products (SOP) macros provide common logic functions by OR gating the outputs of two AND functions or the output of one AND function with one direct input. Variations of inverting and non-inverting inputs are available.

SOP4 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X
SR4CE, SR8CE, SR16CE

4-, 8-, 16-Bit Serial-In Parallel-Out Shift Registers with Clock Enable and Asynchronous Clear

**Architectures Supported**

<table>
<thead>
<tr>
<th>SR4CE, SR8CE, SR16CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SR4CE, SR8CE, and SR16CE are 4-, 8-, and 16-bit shift registers, respectively, with a shift-left serial input (SLI), parallel outputs (Q), and clock enable (CE) and asynchronous clear (CLR) inputs. The CLR input, when High, overrides all other inputs and resets the data outputs (Q) Low. When CE is High and CLR is Low, the data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and CLR is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI → Q0, Q0 → Q1, Q1 → Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the last Q output (Q3 for SR4CE, Q7 for SR8CE, or Q15 for SR16CE) of one stage to the SLI input of the next stage and connecting clock, CE, and CLR in parallel.

The register is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEx, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for SR4CE; z = 7 for SR8CE; z = 15 for SR16CE

qn-1 = state of referenced output one setup time prior to active clock transition

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of sr4ce is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin

process(C, CLR)
begin
    if (CLR='1') then
        Q_INT <= (others => '0');
    elsif (C'event and C='1') then
        if (CE='1') then
            Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
        end if;
    end if;
end process;

Q <= Q_INT;
```

end Behavioral;

Verilog Inference Code

always @ (posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    Q <= {Q[WIDTH-2:0],SL1};
end
SR4CLE, SR8CLE, SR16CLE

4-, 8-, 16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Registers with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>SR4CLE, SR8CLE, SR16CLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SR4CLE, SR8CLE, and SR16CLE are 4-, 8-, and 16-bit shift registers, respectively, with a shift-left serial input (SLI), parallel inputs (D), parallel outputs (Q), and three control inputs: clock enable (CE), load enable (L), and asynchronous clear (CLR). The register ignores clock transitions when L and CE are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q) Low. When L is High and CLR is Low, data on the Dn – D0 inputs is loaded into the corresponding Qn – Q0 bits of the register. When CE is High and L and CLR are Low, data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and CLR are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI → Q0, Q0 → Q1, Q1 → Q2, and so forth).

Registers can be cascaded by connecting the last Q output (Q3 for SR4CLE, Q7 for SR8CLE, or Q15 for SR16CLE) of one stage to the SLI input of the next stage and connecting clock, CE, L, and CLR inputs in parallel.

The register is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEx, or STARTUP_VIRTEx2 symbol.

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of sr4cle is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin

process(C, CLR)
begin
if (CLR='1') then
Q_INT <= (others => '0');
elsif (C'event and C='1') then
z = 3 for SR4CLE; z = 7 for SR8CLE; z = 15 for SR16CLE
qn-1 = state of referenced output one setup time prior to active clock transition

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

X7669
if (CE='1') then
  if (L='1') then
    Q_INT <= D;
  else
    Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
  end if;
end if;
end if;
end if;
end process;

Q <= Q_INT;
end Behavioral;

Verilog Inference Code

always @(posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    if (L)
      Q <= D;
    else
      Q <= {Q[WIDTHT-2:0],SLI};
    end
  end
end
SR4CLED, SR8CLED, SR16CLED

4-, 8-, 16-Bit Shift Registers with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>SR4CLED, SR8CLED, SR16CLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SR4CLED, SR8CLED, and SR16CLED are 4-, 8-, and 16-bit shift registers, respectively, with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D), parallel outputs (Q), and four control inputs: clock enable (CE), load enable (L), shift left/right (LEFT), and asynchronous clear (CLR). The register ignores clock transitions when CE and L are Low. The asynchronous clear, when High, overrides all other inputs and resets the data outputs (Qn) Low. When L is High and CLR is Low, the data on the D inputs is loaded into the corresponding Q bits of the register. When CE is High and L and CLR are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on the SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into the last Q output (Q3 for SR4CLED, Q7 for SR8CLED, or Q15 for SR16CLED) during the Low-to-High clock transition and shifted right (to Q2, Q1,... for SR4CLED; to Q6, Q5,... for SR8CLED; and to Q14, Q13,... for SR16CLED) during subsequent clock transitions. The truth tables for SR4CLED, SR8CLED, and SR16CLED indicate the state of the Q outputs under all input conditions for SR4CLED, SR8CLED, and SR16CLED.

The register is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

SR4CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### SR4CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR L CE LEFT SLI SRI D3 – D0 C</td>
<td>Q0 Q1 Q2 – Q1</td>
</tr>
<tr>
<td>0 0 1 1 SLI X X X</td>
<td>SLI q2 qn-1</td>
</tr>
<tr>
<td>0 0 1 0 X SRI X X</td>
<td>q1 SRI qn+1</td>
</tr>
</tbody>
</table>

$qn-1$ and $qn+1 = state of referenced output one setup time prior to active clock transition

### SR8CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR L CE LEFT SLI SRI D7 – D0 C</td>
<td>Q0 Q1 Q2 – Q1</td>
</tr>
<tr>
<td>1 X X X X X X X</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 X X X X X D7 – D0 X</td>
<td>D0 D7 Dn</td>
</tr>
<tr>
<td>0 0 0 X X X X X</td>
<td>No Chg No Chg No Chg</td>
</tr>
<tr>
<td>0 0 1 1 SLI X X X</td>
<td>SLI q6 qn-1</td>
</tr>
<tr>
<td>0 0 1 0 X SRI X X</td>
<td>q1 SRI qn+1</td>
</tr>
</tbody>
</table>

$qn-1$ or $qn+1 = state of referenced output one setup time prior to active clock transition

### SR16CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR L CE LEFT SLI SRI D15 – D0 C</td>
<td>Q0 Q1 Q2 – Q1</td>
</tr>
<tr>
<td>1 X X X X X X X</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 X X X X X D15 – D0 X</td>
<td>D0 D15 Dn</td>
</tr>
<tr>
<td>0 0 0 X X X X X</td>
<td>No Chg No Chg No Chg</td>
</tr>
<tr>
<td>0 0 1 1 SLI X X X</td>
<td>SLI q14 qn-1</td>
</tr>
<tr>
<td>0 0 1 0 X SRI X X</td>
<td>q1 SRI qn+1</td>
</tr>
</tbody>
</table>

$qn-1$ or $qn+1 = state of referenced output one setup time prior to active clock transition
SR8CLED Implementation Spartan-II, Spartan-IIE, Spartan-3, Virtex, Virtex-E,
Virtex-II, Virtex-II Pro, Virtex-II Pro X

SR8CLED Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of sr4cled is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin

process(C, CLR)
begin
  if (CLR='1') then
    Q_INT <= (others => '0');
  elsif (C'event and C='1') then
    if (CE='1') then
      if (L='1') then
        Q_INT <= D;
      else
        if (LEFT='1') then
          Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
        else
          Q_INT <= SRI & Q_INT(WIDTH-1 downto 1);
        end if;
      end if;
    end if;
  end if;
end if;
end process;

Q <= Q_INT;
end Behavioral;
```

Verilog Inference Code

```verilog
always @(posedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    if (L)
      Q <= D;
    else if (LEFT)
      Q <= {Q[WIDTH-2:0],SLI};
    else
      Q <= {SRI, Q[WIDTH-1:1]};
  end
end
```
SR4RE, SR8RE, SR16RE

4-, 8-, 16-Bit Serial-In Parallel-Out Shift Registers with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>SR4RE, SR8RE, SR16RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XL, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SR4RE, SR8RE, and SR16RE are 4-, 8-, and 16-bit shift registers, respectively, with shift-left serial input (SLI), parallel outputs (Qn), clock enable (CE), and synchronous reset (R) inputs. The R input, when High, overrides all other inputs during the Low-to-High clock (C) transition and resets the data outputs (Q) Low. When CE is High and R is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and R is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI → Q0, Q0 → Q1, Q1 → Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the last Q output (Q3 for SR4RE, Q7 for SR8RE, or Q15 for SR16RE) of one stage to the SLI input of the next stage and connecting clock, CE, and R in parallel.

The register is asynchronously cleared, outputs Low when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

**Inputs**

<table>
<thead>
<tr>
<th>R</th>
<th>CE</th>
<th>SLI</th>
<th>C</th>
<th>Q0</th>
<th>Qz – Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>↑</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>qn-1</td>
</tr>
</tbody>
</table>

SR4RE, SR8RE, and SR16RE are 4-, 8-, and 16-bit shift registers, respectively, with shift-left serial input (SLI), parallel outputs (Qn), clock enable (CE), and synchronous reset (R) inputs. The R input, when High, overrides all other inputs during the Low-to-High clock (C) transition and resets the data outputs (Q) Low. When CE is High and R is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent Low-to-High clock transitions, when CE is High and R is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI → Q0, Q0 → Q1, Q1 → Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the last Q output (Q3 for SR4RE, Q7 for SR8RE, or Q15 for SR16RE) of one stage to the SLI input of the next stage and connecting clock, CE, and R in parallel.

The register is asynchronously cleared, outputs Low when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

**Inputs**

<table>
<thead>
<tr>
<th>R</th>
<th>CE</th>
<th>SLI</th>
<th>C</th>
<th>Q0</th>
<th>Qz – Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>↑</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>qn-1</td>
</tr>
</tbody>
</table>

Usage
For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of sr4re is
  signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin
  process(C)
  begin
    if C'event and C='1' then
      if (R='1') then
        Q_INT <= (others => '0');
      end if;
    end if;
  end process;
end Behavioral;
```

\[
z = 3 \text{ for SR}4\text{RE}; \, z = 7 \text{ for SR}8\text{RE}; \, z = 15 \text{ for SR}16\text{RE}
\]

\[
q_{n-1} = \text{state of referenced output one setup time prior to active clock transition}
\]
```verilog
elsif (CE='1') then
    Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
end if;
end if;
end process;

Q <= Q_INT;
end Behavioral;

Verilog Inference Code
always @ (posedge C)
begin
    if (R)
        Q <= 0;
    else if (CE)
        Q <= (Q[WIDTH-2:0],SLI);
    end
end
```
SR4RLE, SR8RLE, SR16RLE

4-, 8-, 16-Bit Loadable Serial/Parallel-In Parallel-Out Shift Registers with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>SR4RLE, SR8RLE, SR16RLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE Macro</td>
</tr>
<tr>
<td>Spartan-3 Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3 Macro</td>
</tr>
<tr>
<td>CoolRunner-II Macro</td>
</tr>
</tbody>
</table>

SR4RLE, SR8RLE, and SR16RLE are 4-, 8-, and 16-bit shift registers, respectively, with shift-left serial input (SLI), parallel inputs (D), parallel outputs (Q), and three control inputs: clock enable (CE), load enable (L), and synchronous reset (R). The register ignores clock transitions when L and CE are Low. The synchronous R, when High, overrides all other inputs during the Low-to-High clock (C) transition and resets the data outputs (Q) Low. When L is High and R is Low during the Low-to-High clock transition, data on the D inputs is loaded into the corresponding Q bits of the register. When CE is High and L and R are Low, data on the SLI input is loaded into the first bit of the shift register during the Low-to-High clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and R are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI → Q0, Q0 → Q1, Q1 → Q2, and so forth).

Registers can be cascaded by connecting the last Q output (Q3 for SR4RLE, Q7 for SR8RLE, or 15 for SR16RLE) of one stage to the SLI input of the next stage and connecting clock, CE, L, and R inputs in parallel.

The register is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### SR8RLE Implementation


#### Usage

For HDL, these design elements are inferred rather than instantiated.

#### VHDL Inference Code

```vhdl
architecture Behavioral of sr4rle is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin
  process (C)
  begin
    if C'event and C='1' then
      if (R='1') then
        Q0 <= '0'; Qz <= '0'; Q1 <= '0';
      end if;
      if (R='0') then
        Q0 <= '0'; Qz <= '0'; Q1 <= '0';
      end if;
    end if;
  end process;
end Behavioral;
```

---

### Table: SR4RLE, SR8RLE, SR16RLE

<table>
<thead>
<tr>
<th>R</th>
<th>L</th>
<th>CE</th>
<th>SLI</th>
<th>Dz – D0</th>
<th>C</th>
<th>Q0</th>
<th>Qz – Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No Chg</td>
<td>No Chg</td>
</tr>
</tbody>
</table>

- \( z = 3 \) for SR4RLE; \( z = 7 \) for SR8RLE; \( z = 15 \) for SR16RLE
- \( qn-1 \) = state of referenced output one setup time prior to active clock transition

---

![Diagram](attachment:image.png)
Q_INT <= (others => '0');
elsiif (L='1') then
  Q_INT <= D;
elsiif (CE='1') then
  Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
end if;
end if;
end if;
end if;
end if;
end if;
end process;

Q <= Q_INT;

end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (L)
    Q <= D;
  else if (CE)
    Q <= (Q[WIDTH-2:0],SLI);
  end

SR4RLED, SR8RLED, SR16RLED

4-, 8-, 16-Bit Shift Registers with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>SR4RLED, SR8RLED, SR16RLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SR4RLED, SR8RLED, and SR16RLED are 4-, 8-, and 16-bit shift registers, respectively, with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D), parallel outputs (Q) and four control inputs — clock enable (CE), load enable (L), shift left/right (LEFT), and synchronous reset (R). The register ignores clock transitions when CE and L are Low. The synchronous R, when High, overrides all other inputs during the Low-to-High clock (C) transition and resets the data outputs (Q) Low. When L is High and R is Low during the Low-to-High clock transition, the data on the D inputs is loaded into the corresponding Q bits of the register. When CE is High and L and R are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on SLI is loaded into Q0 during the Low-to-High clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into the last Q output (Q3 for SR4RLED, Q7 for SR8RLED, or Q15 for SR16RLED) during the Low-to-High clock transition and shifted right (to Q2, Q1,... for SR4RLED; to Q6, Q5,... for SR8RLED; or to Q14, Q13,... for SR16RLED) during subsequent clock transitions. The truth table indicates the state of the Q outputs under all input conditions.

The register is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net. Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN2, STARTUP_SPARTAN3, STARTUP_VIRTEX, or STARTUP_VIRTEX2 symbol.

SR4RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### SR4RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition

### SR8RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition

### SR16RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition
Usage
For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

architecture Behavioral of sr4rled is

begin

process(C)
begin
  if (C'event and C='1') then
    if (R='1') then
      Q <= (others => '0');
    elsif (CE='1') then
      if (L='1') then
        Q <= D;
      else
        if (LEFT='1') then
          Q <= Q(WIDTH-2 downto 0) & SLI;
        else
          Q <= SRI & Q(WIDTH-1 downto 1);
        end if;
      end if;
    else
      if (LEFT='1') then
        Q <= Q(WIDTH-2 downto 0) & SLI;
      else
        Q <= SRI & Q(WIDTH-1 downto 1);
      end if;
    end if;
  end if;
end process;
end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (R)
    Q <= 0;
  else if (CE)
    if (L)
      Q <= D;
  end if;
end
else if (LEFT)
    Q <= [Q[WIDTH-2:0], SLI];
else
    Q <= [SRI, Q[WIDTH-1:1]];
end
SRD4CE, SRD8CE, SRD16CE

4-, 8-, 16-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Registers with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>SRD4CE, SRD8CE, SRD16CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SRD4CE, SRD8CE, and SRD16CE are 4-, 8-, and 16-bit dual edge triggered shift registers, respectively, with a shift-left serial input (SLI), parallel outputs (Q), clock enable (CE) and asynchronous clear (CLR) inputs. The CLR input, when High, overrides all other inputs and resets the data outputs (Q) Low. When CE is High and CLR is Low, the data on the SLI input is loaded into the first bit of the shift register during the Low-to-High (or High-to-Low) clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and CLR is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the last Q output (Q3 for SRD4CE, Q7 for SRD8CE, or Q15 for SRD16CE) of one stage to the SLI input of the next stage and connecting clock, CE, and CLR in parallel.

The register is asynchronously cleared, outputs Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CE</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for SRD4CE; z = 7 for SRD8CE; z = 15 for SRD16CE
qn-1 = state of referenced output one setup time prior to active clock transition
SRD8CE Implementation CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of srd4ce is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin

process(C, CLR)
begin
  if (CLR='1') then
    Q_INT <= (others => '0');
  elsif (C'event) then
    if (CE='1') then
      Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
    end if;
  end if;
end process;
Q <= Q_INT;
```
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    Q <= (Q[WIDTH-2:0], SL1);
end
SRD4CLE, SRD8CLE, SRD16CLE

4-, 8-, 16-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Registers with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>SRD4CLE, SRD8CLE, SRD16CLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SRD4CLE, SRD8CLE, and SRD16CLE are 4-, 8-, and 16-bit dual edge triggered shift registers, respectively, with a shift-left serial input (SLI), parallel inputs (D), parallel outputs (Q), and three control inputs: clock enable (CE), load enable (L), and asynchronous clear (CLR). The register ignores clock transitions when L and CE are Low. The asynchronous CLR, when High, overrides all other inputs and resets the data outputs (Q) Low. When L is High and CLR is Low, data on the Dn – D0 inputs is loaded into the corresponding Qn – Q0 bits of the register. When CE is High and L and CLR are Low, data on the SLI input is loaded into the first bit of the shift register during the Low-to-High (or High-to-Low) clock (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and L and CLR are Low, the data is shifted to the next highest bit position as new data is loaded into Q0 (SLI→Q0, Q0→Q1, Q1→Q2, and so forth).

Registers can be cascaded by connecting the last Q output (Q3 for SRD4CLE, Q7 for SRD8CLE, or Q15 for SRD16CLE) of one stage to the SLI input of the next stage and connecting clock, CE, L, and CLR inputs in parallel.

The register is asynchronously cleared, outputs Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>CLR</th>
<th>L</th>
<th>CE</th>
<th>SLI</th>
<th>Dn – D0</th>
<th>C</th>
<th>Q0</th>
<th>Qz – Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Dn – D0</td>
<td>↑</td>
<td>D0</td>
<td>Dn</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Dn – D0</td>
<td>↓</td>
<td>D0</td>
<td>Dn</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SLI</td>
<td>X</td>
<td>↑</td>
<td>SLI</td>
<td>qn-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SLI</td>
<td>X</td>
<td>↓</td>
<td>SLI</td>
<td>qn-1</td>
</tr>
</tbody>
</table>
**SRD8CLE Implementation CoolRunner-II**

**Usage**

For HDL, these design elements are inferred rather than instantiated.

**VHDL Inference Code**

```vhdl
architecture Behavioral of srd4cle is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin
process(C, CLR)
begin
if (CLR='1') then
Q_INT <= (others => '0');
elsif (C'event) then
if (CE='1') then
if (L='1') then
Q_INT <= D;
else
Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
end if;
end if;
end if;
end process;
end architecture;
```

\[ z = 3 \text{ for SRD4CLE}; \ z = 7 \text{ for SRD8CLE}; \ z = 15 \text{ for SRD16CLE} \]

qn-1 = state of referenced output one setup time prior to active clock transition
end if;
end if;
end if;
end process;

Q <= Q_INT;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C or posedge CLR)
begin
  if (CLR)
    Q <= 0;
  else if (CE)
    if (L)
      Q <= D;
    else
      Q <= (Q[WIDTH-2:0],SLI);
  end

SRD4CLED, SRD8CLED, SRD16CLED

4-, 8-, 16-Bit Dual Edge Triggered Shift Registers with Clock Enable and Asynchronous Clear

Architectures Supported

<table>
<thead>
<tr>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

SRD4CLED, SRD8CLED, and SRD16CLED are 4-, 8-, and 16-bit dual edge triggered shift registers, respectively, with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D), parallel outputs (Q), and four control inputs: clock enable (CE), load enable (L), shift left/right (LEFT), and asynchronous clear (CLR). The register ignores clock transitions when CE and L are Low. The asynchronous clear, when High, overrides all other inputs and resets the data outputs (Qn) Low. When L is High and CLR is Low, the data on the D inputs is loaded into the corresponding Q bits of the register. When CE is High and L and CLR are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on the SLI is loaded into Q0 during the Low-to-High or High-to-Low clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into the last Q output (Q3 for SRD4CLED, Q7 for SRD8CLED, or Q15 for SRD16CLED) during the Low-to-High or High-to-Low clock transition and shifted right (to Q2, Q1,... for SRD4CLED; to Q6, Q5,... for SRD8CLED; and to Q14, Q13,... for SRD16CLED) during subsequent clock transitions. The truth tables for SRD4CLED, SRD8CLED, and SRD16CLED indicate the state of the Q outputs under all input conditions for SRD4CLED, SRD8CLED, and SRD16CLED.

The register is asynchronously cleared, outputs Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

SRD4CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### SRD4CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 and qn+1 = state of referenced output one setup time prior to active clock transition

### SRD8CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition

### SRD16CLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition
SRD8CLED Implementation CoolRunner-II
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of srd4cled is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin
    process(C, CLR)
    begin
        if (CLR='1') then
            Q_INT <= (others => '0');
        elsif (C'event) then
            if (CE='1') then
                if (L='1') then
                    Q_INT <= D;
                else
                    if (LEFT='1') then
                        Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
                    else
                        Q_INT <= SRI & Q_INT(WIDTH-1 downto 1);
                    end if;
                end if;
            end if;
        end if;
    end if;
end process;

    Q <= Q_INT;
end Behavioral;
```

Verilog Inference Code

```verilog
always @ (posedge C or negedge C or posedge CLR)
begin
    if (CLR)
        Q <= 0;
    else if (CE)
        if (L)
            Q <= D;
        else if (LEFT)
            Q <= {Q[WIDTH-2:0],SLI};
        else
            Q <= {SRI, Q[WIDTH-1:1]};
    end
end
```
SRD4RE, SRD8RE, SRD16RE

4-, 8-, 16-Bit Serial-In Parallel-Out Dual Edge Triggered Shift Registers with Clock Enable and Synchronous Reset

**Architectures Supported**

<table>
<thead>
<tr>
<th>SRD4RE, SRD8RE, SRD16RE</th>
<th>Spartan-II, Spartan-IIE</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>XC9500, XC9500VX, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

SRD4RE, SRD8RE, and SRD16RE are 4-, 8-, and 16-bit dual edge triggered shift registers, respectively, with shift-left serial input (SLI), parallel outputs (Qn), clock enable (CE), and synchronous reset (R) inputs. The R input, when High, overrides all other inputs during the Low-to-High or High-to-Low clock (C) transition and resets the data outputs (Q) Low. When CE is High and R is Low, the data on the SLI is loaded into the first bit of the shift register during the Low-to-High clock or High-to-Low (C) transition and appears on the Q0 output. During subsequent clock transitions, when CE is High and R is Low, data is shifted to the next highest bit position as new data is loaded into Q0 (SLI → Q0, Q0 → Q1, Q1 → Q2, and so forth). The register ignores clock transitions when CE is Low.

Registers can be cascaded by connecting the last Q output (Q3 for SRD4RE, Q7 for SRD8RE, or Q15 for SRD16RE) of one stage to the SLI input of the next stage and connecting clock, CE, and R in parallel.

The register is asynchronously cleared, outputs Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

z = 3 for SRD4RE; z = 7 for SRD8RE; z = 15 for SRD16RE

qn-1 = state of referenced output one setup time prior to active clock transition
Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```
architecture Behavioral of srd4re is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
begin

process(C)
begin
  if C'event then
    if (R='1') then
      Q_INT <= (others => '0');
    elsif (CE='1') then
      Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
    end if;
  end if;
end process;

Q <= Q_INT;
end Behavioral;
```
Verilog Inference Code

always @ (posedge C or negedge C)
begin
  if (R)
    Q <= 0;
  else if (CE)
    Q <= {Q[WIDTH-2:0],SLI};
end
SRD4RLE, SRD8RLE, SRD16RLE

4-, 8-, 16-Bit Loadable Serial/Parallel-In Parallel-Out Dual Edge Triggered Shift Registers with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>SRD4RLE, SRD8RLE, SRD16RLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>
SRD8RLE Implementation CoolRunner-II

Usage

For HDL, these design elements are inferred rather than instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of srd4rle is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
beginn
process(C)
beginn
if (C'event) then
if (R='1') then
Q_INT <= (others => '0');
if (CE='1') then
if (L='1') then
Q_INT <= D;
else
Q_INT <= Q_INT(WIDTH-2 downto 0) & SLI;
end if;
end if;
end if;
end process;
end Behavioral;
```
end if;
end if;
end if;
end if;
end if;
end process;
Q <= Q_INT;
end Behavioral;

Verilog Inference Code

always @ (posedge C or negedge C)
begin
if (R)
    Q <= 0;
else if (CE)
    if (L)
        Q <= D;
    else
        Q <= (Q[WIDTH-2:0],SLI);
end
SRD4RLED, SRD8RLED, SRD16RLED

4-, 8-, 16-Bit Dual Edge Triggered Shift Registers with Clock Enable and Synchronous Reset

Architectures Supported

<table>
<thead>
<tr>
<th>SRD4RLED, SRD8RLED, SRD16RLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SRD4RLED, SRD8RLED, and SRD16RLED are 4-, 8-, and 16-bit dual edge triggered shift registers, respectively, with shift-left (SLI) and shift-right (SRDI) serial inputs, parallel inputs (D), parallel outputs (Q), and four control inputs — clock enable (CE), load enable (L), shift left/right (LEFT), and synchronous reset (R). The register ignores clock transitions when CE and L are Low. The synchronous R, when High, overrides all other inputs during the Low-to-High or High-to-Low clock (C) transition and resets the data outputs (Q) Low. When L is High and R is Low, the data on the D inputs is loaded into the corresponding Q bits of the register. When CE is High and L and R are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on SLI is loaded into Q0 during the Low-to-High or High-to-Low clock transition and shifted left (to Q1, Q2, and so forth) during subsequent clock transitions. If LEFT is Low, data on the SRDI is loaded into the last Q output (Q3 for SRD4RLED, Q7 for SRD8RLED, or Q15 for SRD16RLED) during the Low-to-High or High-to-Low clock transition and shifted right (to Q2, Q1,... for SRD4RLED; to Q6, Q5,... for SRD8RLED; or to Q14, Q13,... for SRD16RLED) during subsequent clock transitions. The truth table indicates the state of the Q outputs under all input conditions.

The register is asynchronously cleared, outputs Low, when power is applied. The power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

SRD4RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
SRD4RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition

SRD8RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition

SRD16RLED Truth Table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition
SRD8RLED Implementation CoolRunner-II
SRL16

16-Bit Shift Register Look-Up-Table (LUT)

Architectures Supported

<table>
<thead>
<tr>
<th>SRL16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
<tr>
<td>No</td>
</tr>
</tbody>
</table>

SRL16 is a shift register look up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. During subsequent Low-to-High clock transitions data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

Static Length Mode

To get a fixed length shift register, drive the A3 through A0 inputs with static values. The length of the shift register can vary from 1 bit to 16 bits as determined from the following formula:

\[
\text{Length} = (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1
\]

If A3, A2, A1, and A0 are all zeros (0000), the shift register is one bit long. If they are all ones (1111), it is 16 bits long.

Dynamic Length Mode

The length of the shift register can be changed dynamically by changing the values driving the A3 through A0 inputs. For example, if A2, A1, and A0 are all ones (111) and A3 toggles between a one (1) and a zero (0), the length of the shift register changes from 16 bits to 8 bits.
Internally, the length of the shift register is always 16 bits and the input lines A3 through A0 select which of the 16 bits reach the output.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am</td>
<td>CLK</td>
</tr>
<tr>
<td>Am</td>
<td>X</td>
</tr>
<tr>
<td>Am</td>
<td>↑</td>
</tr>
</tbody>
</table>

m = 0, 1, 2, 3

Usage

For HDL, this design element can be inferred or instantiated.

**VHDL Inference Code**

```vhdl
architecture Behavioral of srl16 is

signal Q_INT: std_logic_vector(WIDTH-1 downto 0);

begin

process(C)
begin
if (C'event and C='1') then
    Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
end if;
end process;

Q <= Q_INT(WIDTH-1);

end Behavioral;
```

**Verilog Inference Code**

```verilog
always @ (posedge C)
begin
    Q_INT <= {Q_INT[WIDTH-2:0], D};
end

always @(Q_INT)
begin
    Q <= Q_INT[WIDTH-1];
end
```

**VHDL Instantiation Template**

-- Component Declaration for SRL16 should be placed
-- after architecture statement but before begin keyword

cOMPONENT SRL16
-- synthesis translate_off
    generic (INIT: bit_value:= X"0001");
-- synthesis translate_on
    port (Q : out STD_ULOGIC);
```
A0 : in STD_ULOGIC;
A1 : in STD_ULOGIC;
A2 : in STD_ULOGIC;
A3 : in STD_ULOGIC;
CLK : in STD_ULOGIC;
D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRL16
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for SRL16 should be placed
-- in architecture after the begin keyword

SRL16_INSTANCE_NAME : SRL16
-- synthesis translate_off
generic map(
   INIT => hex_value)
-- synthesis translate_on
port map (Q => user_Q,
   A0 => user_A0,
   A1 => user_A1,
   A2 => user_A2,
   A3 => user_A3,
   CLK => user_CLK,
   D => user_D);

Verilog Instantiation Template

SRL16 SRL16_instance_name (.Q (user_Q),
   .A0 (user_A0),
   .A1 (user_A1),
   .A2 (user_A2),
   .A3 (user_A3),
   .CLK (user_CLK),
   .D (user_D));

defparam SRL16_instance_name.INIT = hex_value;

Commonly Used Constraints

BEL, U_SET, INIT
SRL16_1

16-Bit Shift Register Look-Up-Table (LUT) with Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>SRL16_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SRL16_1 is a shift register look up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted. See “Static Length Mode” and “Dynamic Length Mode” in “SRL16”.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am</td>
<td>CLK</td>
</tr>
<tr>
<td>Am</td>
<td>X</td>
</tr>
<tr>
<td>Am</td>
<td>↓</td>
</tr>
</tbody>
</table>

m= 0, 1, 2, 3

Usage

For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of srl16_1 is

signal Q_INT: std_logic_vector(WIDTH-1 downto 0);

begin

process(C)
```

Libraries Guide

ISE 6.li

www.xilinx.com

1-800-255-7778

1629
begin
  if (C'event and C='0') then
    Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
  end if;
end process;

Q <= Q_INT(WIDTH-1);
end Behavioral;

Verilog Inference Code

always @ (negedge C)
begin
  Q_INT <= {Q_INT[WIDTH-2:0],D};
end

always @ (Q_INT)
begin
  Q <= Q_INT[WIDTH-1];
end

VHDL Instantiation Template

-- Component Declaration for SRL16_1 should be placed
-- after architecture statement but before begin keyword

component SRL16_1
  -- synthesis translate_off
  generic (INIT: bit_value:= X"0001");
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    A0 : in STD_ULOGIC;
    A1 : in STD_ULOGIC;
    A2 : in STD_ULOGIC;
    A3 : in STD_ULOGIC;
    CLK : in STD_ULOGIC;
    D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRL16_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for SRL16_1 should be placed
-- in architecture after the begin keyword

SRL16_1_INSTANCE_NAME : SRL16_1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (Q => user_Q,
    A0 => user_A0,
A1 => user_A1,
A2 => user_A2,
A3 => user_A3,
CLK => user_CLK,
D => user_D);

Verilog Instantiation Template

SRL16_1 SRL16_1_instance_name (.Q (user_Q),
    .A0 (user_A0),
    .A1 (user_A1),
    .A2 (user_A2),
    .A3 (user_A3),
    .CLK (user_CLK),
    .D (user_D));

defparam SRL16_1_instance_name.INIT = hex_value;

Commonly Used Constraints

BEL
U_SET
INIT
**SRL16E**

16-Bit Shift Register Look-Up-Table (LUT) with Clock Enable

**Architectures Supported**

<table>
<thead>
<tr>
<th>SRL16E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

SRL16E is a shift register look up table (LUT). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or dynamically adjusted. See “Static Length Mode” and “Dynamic Length Mode” in “SRL16”.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. During subsequent Low-to-High clock transitions, when CE is High, data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

When CE is Low, the register ignores clock transitions.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am</td>
<td>CE</td>
</tr>
<tr>
<td>Am</td>
<td>0</td>
</tr>
<tr>
<td>Am</td>
<td>1</td>
</tr>
</tbody>
</table>

m = 0, 1, 2, 3

**Usage**

For HDL, this design element can be inferred or instantiated.

**VHDL Inference Code**

```vhdl
architecture Behavioral of srl16e is

signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
```
begin

process(C)
begin
  if (C'event and C='1') then
    if (CE='1') then
      Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
    end if;
  end if;
end process;

Q <= Q_INT(WIDTH-1);
end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (CE)
    Q_INT <= {Q_INT[WIDTH-2:0],D};
  end

always @(Q_INT)
begin
  Q <= Q_INT[WIDTH-1];
end

VHDL Instantiation Template

-- Component Declaration for SRL16E should be placed
-- after architecture statement but before begin keyword

component SRL16E
  -- synthesis translate_off
  generic (INIT: bit_value:= X"0001");
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
       A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A3 : in STD_ULOGIC;
       CE : in STD_ULOGIC;
       CLK : in STD_ULOGIC;
       D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRL16E
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for SRL16E should be placed
-- in architecture after the begin keyword
SRL16E_INSTANCE_NAME : SRL16E
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            A0 => user_A0,
            A1 => user_A1,
            A2 => user_A2,
            A3 => user_A3,
            CE => user_CE,
            CLK => user_CLK,
            D => user_D);

Verilog Instantiation Template

SRL16E SRL16E_instance_name (.Q (user_Q),
                             .A0 (user_A0),
                             .A1 (user_A1),
                             .A2 (user_A2),
                             .A3 (user_A3),
                             .CE (user_CE),
                             .CLK (user_CLK),
                             .D (user_D));

defparam SRL16E_instance_name.INIT = hex_value;

Commonly Used Constraints

BEL
U_SET
INIT
SRL16E_1

16-Bit Shift Register Look-Up-Table (LUT) with Negative-Edge Clock and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>SRLC16E_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

SRL16E_1 is a shift register look up table (LUT) with clock enable (CE). The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or dynamically adjusted. See “Static Length Mode” and “Dynamic Length Mode” in the “SRL16”.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions, when CE is High, data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

When CE is Low, the register ignores clock transitions.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am 0</td>
<td>CE 0</td>
</tr>
<tr>
<td>Am 1</td>
<td>CE 1</td>
</tr>
</tbody>
</table>

m= 0, 1, 2, 3

Usage

For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

```
architecture Behavioral of srl16e_1 is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);
```
begin

process(C)
begin
if (C'event and C='0') then
  if (CE='1') then
    Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
  end if;
end if;
end if;
end process;

Q <= Q_INT(WIDTH-1);
end Behavioral;

Verilog Inference Code
always @ (negedge C)
begin
if (CE)
  Q_INT <= {Q_INT[WIDTH-2:0],D};
end

always @(Q_INT)
begin
  Q <= Q_INT[WIDTH-1];
end

VHDL Instantiation Template
-- Component Declaration for SRL16E_1 should be placed
-- after architecture statement but before begin keyword

component SRL16E_1
  -- synthesis translate_off
  generic (INIT : bit_value := X"0001");
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
        A0 : in STD_ULOGIC;
        A1 : in STD_ULOGIC;
        A2 : in STD_ULOGIC;
        A3 : in STD_ULOGIC;
        CE : in STD_ULOGIC;
        CLK : in STD_ULOGIC;
        D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRL16E_1
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for SRL16E_1 should be placed
-- in architecture after the begin keyword
SRL16E_1_INSTANCE_NAME : SRL16E_1
-- synthesis translate_off
generic map (INIT => hex_value)
-- synthesis translate_on
port map (Q => user_Q,
    A0 => user_A0,
    A1 => user_A1,
    A2 => user_A2,
    A3 => user_A3,
    CE => user_CE,
    CLK => user_CLK,
    D => user_D);

Verilog Instantiation Template

SRL16E_1 SRL16E_1_instance_name (.Q (user_Q),
    .A0 (user_A0),
    .A1 (user_A1),
    .A2 (user_A2),
    .A3 (user_A3),
    .CE (user_CE),
    .CLK (user_CLK),
    .D (user_D));

defparam SRL16E_1_instance_name.INIT = hex_value;

Commonly Used Constraints

BEL
U_SET
INIT
SRLC16

16-Bit Shift Register Look-Up-Table (LUT) with Carry

Architectures Supported

<table>
<thead>
<tr>
<th>SRLC16</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

SRLC16 is a shift register look up table (LUT) with Carry. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length, or it may be dynamically adjusted.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. During subsequent Low-to-High clock transitions data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

For information about the static length mode, see “Static Length Mode” in “SRL16”.

For information about the dynamic length mode, see “Dynamic Length Mode” in “SRL16”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_m )</td>
<td>CLK</td>
</tr>
<tr>
<td>( A_m )</td>
<td>X</td>
</tr>
<tr>
<td>( A_m )</td>
<td>( \uparrow )</td>
</tr>
</tbody>
</table>

\( m= 0, 1, 2, 3 \)

Usage

For HDL, this design element can be inferred or instantiated.
VHDL Inference Code

architecture Behavioral of srlc16 is

signal Q_INT: std_logic_vector(WIDTH-1 downto 0);

begin

process(C)
begin
if (C'event and C='1') then
  Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
end if;
end process;

Q <= Q_INT(WIDTH-1);
end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  Q_INT <= {Q_INT[WIDTH-2:0],D};
end

always @(Q_INT)
begin
  Q <= Q_INT[WIDTH-1];
end

VHDL Instantiation Template

-- Component Declaration for SRLC16 should be placed
-- after architecture statement but before begin keyword

component SRLC16
-- synthesis translate_off
  generic (INIT : bit_value := X"0001");
-- synthesis translate_on
  port (Q : out STD_ULOGIC;
      Q15 : out STD_ULOGIC;
      A0 : in STD_ULOGIC;
      A1 : in STD_ULOGIC;
      A2 : in STD_ULOGIC;
      A3 : in STD_ULOGIC;
      CLK : in STD_ULOGIC;
      D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRLC16
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section
-- Component Instantiation for SRLC16 should be placed
-- in architecture after the begin keyword

SRLC16_INSTANCE_NAME : SRLC16
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (Q => user_Q,
            Q15 => user_Q15,
            A0 => user_A0,
            A1 => user_A1,
            A2 => user_A2,
            A3 => user_A3,
            CLK => user_CLK,
            D => user_D);

Verilog Instantiation Template

SRLC16 SRLC16_instance_name (.Q (user_Q),
                            .Q15 (user_Q15),
                            .A0 (user_A0),
                            .A1 (user_A1),
                            .A2 (user_A2),
                            .A3 (user_A3),
                            .CLK (user_CLK),
                            .D (user_D));

defparam SRLC16_instance_name.INIT = hex_value;

Commonly Used Constraints

INIT
SRLC16_1

16-Bit Shift Register Look-Up-Table (LUT) with Carry and Negative-Edge Clock

Architectures Supported

<table>
<thead>
<tr>
<th>SRLC16_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

SRLC16_1 is a shift register look up table (LUT) with carry and a negative-edge clock. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted. See “Static Length Mode” and “Dynamic Length Mode” in “SRL16”.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am X</td>
<td>Q(Am)</td>
</tr>
<tr>
<td>Am D</td>
<td>Q(Am-1)</td>
</tr>
</tbody>
</table>

m= 0, 1, 2, 3

Usage

For HDL, this design element can inferred.

VHDL Inference Code

```vhdl
architecture Behavioral of srlc16_1 is

signal Q_INT: std_logic_vector(WIDTH-1 downto 0);

begin
```
process(C)
begin
  if (C'event and C='0') then
    Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
  end if;
end process;

Q <= Q_INT(WIDTH-1);
end Behavioral;

Verilog Inference Code
always @(negedge C)
begin
  Q_INT <= {Q_INT[WIDTH-2:0], D};
end

always @(Q_INT)
begin
  Q <= Q_INT(WIDTH-1);
end
SRLC16E

16-Bit Shift Register Look-Up-Table (LUT) with Carry and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th>SRLC16E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

SRLC16E is a shift register look up table (LUT) with carry and clock enable. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

The data (D) is loaded into the first bit of the shift register during the Low-to-High clock (CLK) transition. When CE is High, during subsequent Low-to-High clock transitions, data is shifted to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

For information about the static length mode, see “Static Length Mode” in “SRL16”.

For information about the dynamic length mode, see “Dynamic Length Mode” in “SRL16”.

<table>
<thead>
<tr>
<th>Am</th>
<th>CLK</th>
<th>CE</th>
<th>D</th>
<th>Q</th>
<th>Q15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>Q(Am)</td>
<td>Q(15)</td>
</tr>
<tr>
<td>Am</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>Q(Am)</td>
<td>Q(15)</td>
</tr>
<tr>
<td>Am</td>
<td>↑</td>
<td>1</td>
<td>D</td>
<td>Q(Am-1)</td>
<td>Q15</td>
</tr>
</tbody>
</table>

m= 0, 1, 2, 3

Usage

For HDL, this design element can be inferred or instantiated.
VHDL Inference Code

architecture Behavioral of srlc16e is

signal Q_INT: std_logic_vector(WIDTH-1 downto 0);

begin

process(C)
begin
  if (C'event and C='1') then
    if (CE='1') then
      Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
    end if;
  end if;
end process;

Q <= Q_INT(WIDTH-1);

end Behavioral;

Verilog Inference Code

always @ (posedge C)
begin
  if (CE)
    Q_INT <= {Q_INT[WIDTH-2:0],D};
end

always @ (Q_INT)
begin
  Q <= Q_INT[WIDTH-1];
end

VHDL Instantiation Template

-- Component Declaration for SRLC16E should be placed
-- after architecture statement but before begin keyword

component SRLC16E
  -- synthesis translate_off
  generic (INIT : bit_value := X"0001");
  -- synthesis translate_on
  port (Q : out STD_ULOGIC;
    Q15 : out STD_ULOGIC;
    A0 : in STD_ULOGIC;
    A1 : in STD_ULOGIC;
    A2 : in STD_ULOGIC;
    A3 : in STD_ULOGIC;
    CE : in STD_ULOGIC;
    CLK : in STD_ULOGIC;
    D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRLC16E
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes in this section

-- Component Instantiation for SRLC16E should be placed
-- in architecture after the begin keyword

SRLC16E_INSTANCE_NAME : SRLC16E
   -- synthesis translate_off
   generic map (INIT => hex_value)
   -- synthesis translate_on
   port map (Q => user_Q,
            Q15 => user_Q15,
            A0 => user_A0,
            A1 => user_A1,
            A2 => user_A2,
            A3 => user_A3,
            CE => user_CE,
            CLK => user_CLK,
            D => user_D);

Verilog Instantiation Template

SRLC16E SRLC16E_instance_name (.Q (user_Q),
                               .Q15 (user_Q15),
                               .A0 (user_A0),
                               .A1 (user_A1),
                               .A2 (user_A2),
                               .A3 (user_A3),
                               .CE (user_CE),
                               .CLK (user_CLK),
                               .D (user_D));

defparam SRLC16E_instance_name.INIT = hex_value;

Commonly Used Constraints

INIT
SRLC16E_1

16-Bit Shift Register Look-Up-Table (LUT) with Carry, Negative-Edge Clock, and Clock Enable

Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>SRLC16E_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II,</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-IIIE</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex,</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-E</td>
<td></td>
</tr>
<tr>
<td>Virtex-II,</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II Pro</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II Pro</td>
<td></td>
</tr>
<tr>
<td>Xilinx 9500,</td>
<td>No</td>
</tr>
<tr>
<td>Xilinx 9500XV</td>
<td></td>
</tr>
<tr>
<td>Xilinx 9500XL</td>
<td></td>
</tr>
<tr>
<td>CoolRunner</td>
<td>No</td>
</tr>
<tr>
<td>XPLA3</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

SRLC16E_1 is a shift register look up table (LUT) with carry, clock enable, and negative-edge clock. The inputs A3, A2, A1, and A0 select the output length of the shift register. The shift register may be of a fixed, static length or it may be dynamically adjusted. See “SRLC16” and “Dynamic Length Mode” in “SRL16”.

The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the High-to-Low clock (CLK) transition. During subsequent High-to-Low clock transitions data is shifted to the next highest bit position as new data is loaded when CE is High. The data appears on the Q output when the shift register length determined by the address inputs is reached.

The Q15 output is available for the user to cascade multiple shift register LUTs to create larger shift registers.

<table>
<thead>
<tr>
<th>Am</th>
<th>CE</th>
<th>CLK</th>
<th>D</th>
<th>Q</th>
<th>Q15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Am</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Q(Am)</td>
<td>No Chg</td>
</tr>
<tr>
<td>Am</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Q(Am)</td>
<td>No Chg</td>
</tr>
<tr>
<td>Am</td>
<td>1</td>
<td>↓</td>
<td>D</td>
<td>Q(Am-1)</td>
<td>Q14</td>
</tr>
</tbody>
</table>

m= 0, 1, 2, 3

Usage

For HDL, this design element can be inferred or instantiated.

VHDL Inference Code

architecture Behavioral of srlc16e_1 is
signal Q_INT: std_logic_vector(WIDTH-1 downto 0);

begin

process(C)
begin
if (C'event and C='0') then
if (CE='1') then
Q_INT <= Q_INT(WIDTH-2 downto 0) & D;
end if;
end if;
end process;

Q <= Q_INT(WIDTH-1);
end Behavioral;

Verilog Inference Code
always @ (negedge C)
begin
if (CE)
Q_INT <= {Q_INT[WIDTH-2:0],D};
end

always @(Q_INT)
begin
Q <= Q_INT[WIDTH-1];
end

VHDL Instantiation Template
-- Component Declaration for SRLC16E_1 should be placed
-- after architecture statement but before begin keyword

component SRLC16E_1
-- synthesis translate_off
generic (INIT : bit_value := X"0001");
-- synthesis translate_on
port (Q : out STD_ULOGIC;
Q15 : out STD_ULOGIC;
A0 : in STD_ULOGIC;
A1 : in STD_ULOGIC;
A2 : in STD_ULOGIC;
A3 : in STD_ULOGIC;
CE : in STD_ULOGIC;
CLK : in STD_ULOGIC;
D : in STD_ULOGIC);
end component;

-- Component Attribute specification for SRLC16E_1
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes in this section
-- Component Instantiation for SRLC16E_1 should be placed
-- in architecture after the begin keyword

SRLC16E_1_INSTANCE_NAME : SRLC16E_1
  -- synthesis translate_off
  generic map (INIT => hex_value)
  -- synthesis translate_on
  port map (Q => user_Q,
           Q15 => user_Q15,
           A0 => user_A0,
           A1 => user_A1,
           A2 => user_A2,
           A3 => user_A3,
           CE => user_CE,
           CLK => user_CLK,
           D => user_D);

Verilog Instantiation Template

SRLC16E_1 SRLC16E_1_instance_name (.Q (user_Q),
                        .Q15 (user_Q15),
                        .A0 (user_A0),
                        .A1 (user_A1),
                        .A2 (user_A2),
                        .A3 (user_A3),
                        .CE (user_CE),
                        .CLK (user_CLK),
                        .D (user_D));

defparam SRLC16E_1_instance_name.INIT = hex_value;

Commonly Used Constraints

INIT
**STARTBUF_architecture**

VHDL Simulation of FPGA Designs

**Architectures Supported**

<table>
<thead>
<tr>
<th>STARTBUF_architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

STARTBUF_architecture is used for VHDL simulation of FPGA designs that require the use of the STARTUP block. The difference between the STARTBUF_architecture and the STARTUP block is that the STARTBUF_architecture contains output ports which may be connected to all register set/resets in the design (GSROUT) or to all I/O three-state controls (GTSOUT) so that these functions may be functionally simulated. This design element should not be used for Verilog or schematic entry. In order to use the STARTBUF_architecture, the desired input(s) should be connected to a top-level port in the design and the corresponding output(s) must be connected to either the three-state control signal for all inferred and instantiated output buffers in the design (GTSOUT) or all inferred or instantiated register set/resets in the design.

During simulation, the inputs to the STARTBUF_architecture can be toggled by the testbench in order to activate the global three-state or global set/reset signal in the design. This should be done at the beginning of the simulation to simulate the behavior of the registers and I/O during configuration. It may also be applied during simulation to simulate a reconfiguration (ProG pin high) of the device. During synthesis and implementation, this component will be treated as a STARTUP block. The connected input ports to this component should remain in the design and be connected to the correct corresponding global resource.

For more information, see the Xilinx Synthesis and Verification Design Guide.

The value at port GSROUT will be always the be value at port GSRIN. The value at port GTSOUT will always be the value at port GTSIN. CLKin has no effect on simulation.

**VHDL Instantiation Code**

Following are three examples:

```vhdl
component STARTBUF_SPARTAN2
    port (GSROUT : out std_ulogic;
          GTSOUT : out std_ulogic;
          CLKin : in std_ulogic;
          GSRIN : in std_ulogic;
          GTSIN : in std_ulogic);
end component;
```
component STARTBUF_VIRTEX
  port (GTSOUT : out std_ulogic;
        GSROUT : out std_ulogic;
        CLKIN  : in  std_ulogic;
        GSRIN  : in  std_ulogic;
        GTSIN  : in  std_ulogic);
end component;

component STARTBUF_VIRTEX2
  port (GSROUT : out std_ulogic;
        GTSOUT : out std_ulogic;
        CLKIN  : in  std_ulogic;
        GSRIN  : in  std_ulogic;
        GTSIN  : in  std_ulogic);
end component;

Commonly Used Constraints

None
Spartan-II User Interface to Global Clock, Reset, and 3-State Controls

### Architectures Supported

<table>
<thead>
<tr>
<th>STARTUP_SPARTAN2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>Primitive*</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>No</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

*Supported for Spartan-II but not for Spartan-IIE, which is supported by STARTUP_VIRTEX.

The STARTUP_SPARTAN2 primitive is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets all flip-flops, all latches, and every block RAM (RAMB4) output register in the device, depending on the initialization state (S or R) of the component.

**Note:** Block RAMB4 content, LUT RAMs, delay locked loop elements (CLKDLL, CLKDLLHF, BUFGDLL), and shift register LUTs (SRL16, SRL16_1, SRL16E, SRL16E_1) are not set/reset.

Following configuration, the global 3-state control (GTS), when High—and BSCAN is not enabled and executing an EXTEST instruction—forces all the IOB outputs into high impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

**Note:** GTS= Global 3-State

Including the STARTUP_SPARTAN2 symbol in a design is optional. You must include the symbol under the following conditions.

- To exert external control over global set/reset, connect the GSR pin to a top level port and an IBUF, as shown here.

- To exert external control over global 3-state, connect the GTS pin to a top level port and IBUF, as shown here.
To synchronize startup to a user clock, connect the user clock signal to the CLK input, as shown here. Furthermore, “user clock” must be selected in the BitGen program.

You can use location constraints to specify the pin from which GSR or GTS (or both) is accessed.

Usage
For HDL, this design element typically is instantiated rather than inferred.

VHDL Instantiation Template
-- Component Declaration for STARTUP_SPARTAN2 should be placed
-- after architecture statement but before begin keyword

component STARTUP_SPARTAN2
  port (CLK : in STD_ULOGIC;
       GSR : in STD_ULOGIC;
       GTS : in STD_ULOGIC);
end component;

-- Component Attribute specification for STARTUP_SPARTAN2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for STARTUP_SPARTAN2 should be placed
-- in architecture after the begin keyword

STARTUP_SPARTAN2_INSTANCE_NAME : STARTUP_SPARTAN2
port map (CLK => user_CLK,
        GSR => user_GSR,
        GTS => user_GTS);

Verilog Instantiation Template
STARTUP_SPARTAN2 STARTUP_SPARTAN2_instance_name (.CLK (user_CLK),
                  .GSR (user_GSR),
                  .GTS (user_GTS));

Commonly Used Constraints
None
STARTUP_SPARTAN3

Spartan-3 User Interface to Global Clock, Reset, and 3-State Controls

Architectures Supported

<table>
<thead>
<tr>
<th>STARTUP_SPARTAN3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

The STARTUP_SPARTAN3 primitive is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets all flip-flops, all latches, and every block RAMB16 output register in the device, depending on the initialization state (INIT=1 or 0) of the component.

**Note:** Block RAM content, LUT RAMs, the Digital Clock Manager (DCM), and shift register LUTs (SRL16, SRL16_1, SRL16E, SRL16E_1, SRLC16, SRLC16_1, SRLC16E, and SRLC16E_1) are not set/reset.

Following configuration, the global 3-state control (GTS), when High—and BSCAN is not enabled and executing an EXTEST instruction—forces all the IOB outputs into high impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

**Note:** GTS= Global 3-State

Including the STARTUP_SPARTAN3 symbol in a design is optional. You must include the symbol under the following conditions.

- To exert external control over global set/reset, connect the GSR pin to a top level port and an IBUF, as shown here.

- To exert external control over global 3-state, connect the GTS pin to a top level port and IBUF, as shown here.
To synchronize startup to a user clock, connect the user clock signal to the CLK input, as shown here. Furthermore, “user clock” must be selected in the BitGen program.

You can use location constraints to specify the pin from which GSR or GTS (or both) is accessed.

Usage

For HDL, this design element typically is instantiated rather than inferred.

VHDL Instantiation Template

```vhdl
-- Component Declaration for STARTUP_SPARTAN3 should be placed
-- after architecture statement but before begin keyword

component STARTUP_SPARTAN3
    port (CLK : in STD_ULOGIC;
          GSR : in STD_ULOGIC;
          GTS : in STD_ULOGIC);
end component;

-- Component Attribute specification for STARTUP_SPARTAN3
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for STARTUP_SPARTAN3 should be placed
-- in architecture after the begin keyword

STARTUP_SPARTAN3_INSTANCE_NAME : STARTUP_SPARTAN3
    port map (CLK => user_CLK,
              GSR => user_GSR,
              GTS => user_GTS);
```

Verilog Instantiation Template

```verilog
STARTUP_SPARTAN3 STARTUP_SPARTAN3_instance_name (.CLK (user_CLK),
                                                  GSR (user_GSR),
                                                  ...)```
Commonly Used Constraints

None.
Virtex and Virtex-E User Interface to Global Clock, Reset, and 3-State Controls

Architectures Supported

<table>
<thead>
<tr>
<th>STARTUP_VIRTEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

*Supported for Spartan-IIIE but not for Spartan-II which is supported by STARTUP_SPARTAN2.

The STARTUP_VIRTEX primitive is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets all flip-flops, all latches, and every block RAM (RAMB4) output register in the device, depending on the initialization state (S or R) of the component. For Virtex-II, Virtex-II Pro, and Virtex-II Pro X, see “STARTUP_VIRTEX2”.

**Note:** Block RAMB4 content, LUT RAMs, delay locked loop elements (CLKDLL, CLKDLLHF, BUFGDLL), and shift register LUTs (SRL16, SRL16_1, SRL16E, SRL16E_1) are not set/reset.

Following configuration, the global 3-state control (GTS), when High—and BSCAN is not enabled and executing an EXTEST instruction—forces all the IOB outputs into high impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

**Note:** GTS = Global 3-State

Including the STARTUP_VIRTEX symbol in a design is optional. You must include the symbol under the following conditions.

- To exert external control over global set/reset, connect the GSR pin to a top level port and an IBUF, as shown here.

- To exert external control over global 3-state, connect the GTS pin to a top level port and IBUF, as shown here.
To synchronize startup to a user clock, connect the user clock signal to the CLK input, as shown here. Furthermore, “user clock” must be selected in the BitGen program.

You can use location constraints to specify the pin from which GSR or GTS (or both) is accessed.

**Usage**

For HDL, this design element typically is instantiated rather than inferred.

**VHDL Instantiation Template**

```
-- Component Declaration for STARTUP_VIRTEX should be placed
-- after architecture statement but before begin keyword

component STARTUP_VIRTEX
    port (CLK : in STD_ULOGIC;
        GSR : in STD_ULOGIC;
        GTS : in STD_ULOGIC);
end component;

-- Component Attribute specification for STARTUP_VIRTEX
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for STARTUP_VIRTEX should be placed
-- in architecture after the begin keyword

STARTUP_VIRTEX_INSTANCE_NAME : STARTUP_VIRTEX
    port map (CLK => user_CLK,
        GSR => user_GSR,
        GTS => user_GTS);
```

**Verilog Instantiation Template**

```
STARTUP_VIRTEX STARTUP_VIRTEX_instance_name (.CLK (user_CLK),
    .GSR (user_GSR),
    .GTS (user_GTS));
```
Commonly Used Constraints

None
Virtex-II, Virtex-II Pro, and Virtex-II Pro X User Interface to Global Clock, Reset, and 3-State Controls

Architectures Supported

<table>
<thead>
<tr>
<th>STARTUP_VIRTEX2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td>No</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>No</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>No</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X Primitive</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The STARTUP_VIRTEX2 primitive is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets all flip-flops, all latches, and every block RAMB16 output register in the device, depending on the initialization state (INIT=1 or 0) of the component. For Virtex and Virtex-E, see “STARTUP_VIRTEX”.

**Note:** Block RAM content, LUT RAMs, the Digital Clock Manager (DCM), and shift register LUTs (SRL16, SRL16_1, SRL16E, SRL16E_1, SRLC16, SRLC16_1, SRLC16E, and SRLC16E_1) are not set/reset.

Following configuration, the global 3-state control (GTS), when High—and BSCAN is not enabled and executing an EXTEST instruction—forces all the IOB outputs into high impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

**Note:** GTS= Global 3-State

Including the STARTUP_VIRTEX2 symbol in a design is optional. You must include the symbol under the following conditions.

- To exert external control over global set/reset, connect the GSR pin to a top level port and an IBUF, as shown here.

- To exert external control over global 3-state, connect the GTS pin to a top level port and IBUF, as shown here.
To synchronize startup to a user clock, connect the user clock signal to the CLK input, as shown here. Furthermore, “user clock” must be selected in the BitGen program.

You can use location constraints to specify the pin from which GSR or GTS (or both) is accessed.

Usage

For HDL, this design element typically is instantiated rather than inferred.

**VHDL Instantiation Template**

```vhdl
-- Component Declaration for STARTUP_VIRTEX2 should be placed
-- after architecture statement but before begin keyword

component STARTUP_VIRTEX2
    port (CLK : in STD_ULOGIC;
          GSR : in STD_ULOGIC;
          GTS : in STD_ULOGIC);
end component;

-- Component Attribute specification for STARTUP_VIRTEX2
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes in this section

-- Component Instantiation for STARTUP_VIRTEX2 should be placed
-- in architecture after the begin keyword

STARTUP_VIRTEX2_INSTANCE_NAME : STARTUP_VIRTEX2
    port map (CLK => user_CLK,
              GSR => user_GSR,
              GTS => user_GTS);
```

Verilog Instantiation Template

STARTUP_VIRTEX2 STARTUP_VIRTEX2_instance_name (.CLK (user_CLK),
    GSR (user_GSR),
    GTS (user_GTS));

Commonly Used Constraints

None
Three-State On Configuration

Architectures Supported

<table>
<thead>
<tr>
<th>TOC</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

The TOC is a component used for VHDL simulation of FPGA designs. This component should not be used for Verilog or schematic entry. The TOC’s function is to mimic the function of the internal three-state signal during the FPGA configuration process. In order to use TOC, it must be connected to the three-state signal for all inferred and instantiated output buffers in the design. During synthesis and implementation, this three-state signal will use the dedicated global three-state network and will not use local routing resources. During simulation, TOC will emit a one-shot pulse for the amount of time specified by the WIDTH generic (default is 100 ns). This one-shot pulse is intended to three-state all outputs so that at the beginning of operation, all outputs are not being driven as would happen in the real silicon during configuration of the device.

For more information, see the Xilinx Synthesis and Verification Design Guide.

Port O will be high at simulation time 0 for the amount of time specified by the WIDTH generic attribute. After that time, it will be 0. This will not affect implementation in any way.

VHDL Instantiation Code

```vhdl
component TOC
  -- synthesis translate_off
  generic (WIDTH : Time := 100 ns);
  -- synthesis translate_on
  port (O : out std_ulogic := '0');
end component;
```

Commonly Used Constraints

For simulation, the WIDTH generic can be modified to change the amount of time the one-shot pulse is applied for.

There are no supported constraints for this component for implementation.
Three-State On Configuration Buffer

Architectures Supported

<table>
<thead>
<tr>
<th>TOCBUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

The TOCBUF is a component used for VHDL simulation of FPGA designs. It is similar to the TOC component except that it contains an input for controlling the global I/O three-state function rather than a one-shot. This component should not be used for Verilog or schematic entry. The TOCBUF’s function allows user control of the function of the global I/O three-state signal as done during the FPGA configuration process. In order to use the TOCBUF, the input should be connected to a top-level port in the design and the output must be connected to the three-state control signal for all inferred and instantiated output buffers in the design.

During simulation, the input to the TOCBUF can be toggled by the testbench in order to activate the global three-state signal in the design. This should be done at the beginning of the simulation to simulate the behavior of the I/O during configuration. It may also be applied during simulation to simulate a reconfiguration (ProG pin high) of the device. During synthesis and implementation, this three-state signal uses the dedicated global three-state network and does not use local routing resources. The port connected to this component is optimized out of the design and does not use any pin resources. If you want to have the port implemented in the design, a STARTBUF_architecture should be used. In order to replace this port during back-end simulation, the -tp switch should be used when invoking the NGD2VER or NGD2VHDL netlister. If using the ISE GUI, use the Bring Out Global Three-state Net as a Port option in the Simulation Model Properties window.

For more information, see the Xilinx Synthesis and Verification Design Guide.

The value at port O will be always be the value at port I (it is a buffer).

VHDL Instantiation Code

```vhdl
component TOCBUF
    port (I : in std_ulogic;
          O : out std_ulogic);
end component;
```

Commonly Used Constraints

None
VCC

VCC-Connection Signal Tag

Architectures Supported

<table>
<thead>
<tr>
<th>VCC</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
<td></td>
</tr>
<tr>
<td>Spartan-3</td>
<td></td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td></td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro</td>
<td></td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td></td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td></td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td></td>
</tr>
</tbody>
</table>

The VCC signal tag or parameter forces a net or input function to a logic High level. A net tied to VCC cannot have any other source.

When the placement and routing software encounters a net or input function tied to VCC, it removes any logic that is disabled by the VCC signal. The VCC signal is only implemented when the disabled logic cannot be removed.

Usage

For HDL, this design element can be instantiated or inferred.

**VHDL Inference Code**

```
vcc_signal <= '1';
```

**Verilog Inference Code**

```
assign vcc_signal = 1;
```

**VHDL Instantiation Template**

```
-- Component Declaration for VCC should be placed
-- after architecture statement but before begin keyword

component VCC
  port (P : out STD_ULOGIC);
end component;

-- Component Attribute specification for VCC
-- should be placed after architecture declaration but
-- before the begin keyword

-- Enter attributes here

-- Component Instantiation for VCC should be placed
-- in architecture after the begin keyword
```
VCC_INSTANCE_NAME : VCC
    port map (P => user_P);

Verilog Instantiation Template

VCC instance_name (.P (user_P));

Commonly Used Constraints

None
UPAD

Connects the I/O Node of an IOB to the Internal PLD Circuit

Architectures Supported

<table>
<thead>
<tr>
<th>UPAD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

A UPAD allows the use of any unbonded IOBs in a device. It is used the same way as an IOPAD except that the signal output is not visible on any external device pins.
# XNOR2-9

## 2- to 9-Input XNOR Gates with Non-Inverted Inputs

### Architectures Supported

<table>
<thead>
<tr>
<th></th>
<th>XNOR2, XNOR3, XNOR4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Primitive</td>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Primitive</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>XNOR5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>XNOR6, XNOR7, XNOR8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>XNOR9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Macro</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Macro</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Macro</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>Macro</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>Macro</td>
</tr>
</tbody>
</table>
XNOR Gate Representations

XNOR functions of up to nine inputs are available. All inputs are non-inverting. Because each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.

XNOR5 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

XNOR6 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
XNOR2-9

XNOR7 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

XNOR7 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

XNOR7 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

XNOR8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
XNOR8 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E

XNOR8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

XNOR9 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II
XNOR9 Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E

Usage
For HDL, these design elements can be inferred or instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of xnor2 is
begin
  process (I0, I1)
  begin
    O <= I0 xnor I1;
  end process;
end Behavioral;
```
Verilog Inference Code

always @ (I0 or I1)
begin
    O <= I0~^I1;
end

VHDL Instantiation Template for XNOR5

Following is the VHDL code for XNOR5. To instantiate XNOR2, remove I2, I3, and I4. To instantiate XNOR3, remove I3 and I4. For XNOR4, remove I4. XNOR2B1, and XNOR2B2 have the same code as XNOR2. XNOR3B1, 3B2, and 3B3 have the same code as XNOR3 and so forth.

-- Component Declaration for XNOR5 should be placed
-- after architecture statement but before begin keyword

cOMPONENT XNOR5
    PORT (O : OUT STD_ULOGIC;
          I0 : IN STD_ULOGIC;
          I1 : IN STD_ULOGIC;
          I2 : IN STD_ULOGIC;
          I3 : IN STD_ULOGIC;
          I4 : IN STD_ULOGIC);
END COMPONENT;

-- Component Attribute specification for XNOR5
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for XNOR5 should be placed
-- in architecture after the begin keyword

XNOR5_INSTANCE_NAME : XNOR5
    PORT MAP (O => user_O,
               I0 => user_I0,
               I1 => user_I1,
               I2 => user_I2,
               I3 => user_I3,
               I4 => user_I4);

Verilog Instantiation Template for XNOR5

XNOR5 XNOR5_instance_name (.O (user_O),
                           .i0 (user_I0),
                           .i1 (user_I1),
                           .i2 (user_I2),
                           .i3 (user_I3),
                           .i4 (user_I4));

Commonly Used Constraints

None
2- to 9-Input XOR Gates with Non-Inverted Inputs

Architectures Supported

<table>
<thead>
<tr>
<th>XOR2, XOR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XOR4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XOR5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XOR6, XOR7, XOR9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIE</td>
</tr>
<tr>
<td>Spartan-3</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
</tr>
<tr>
<td>CoolRunner-II</td>
</tr>
</tbody>
</table>
**XOR Gate Representations**

XOR functions of up to nine inputs are available. All inputs are non-inverting. Because each input uses a CLB resource, replace functions with unused inputs with functions having the necessary number of inputs.
XOR6 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

XOR7 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

XOR8 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

XOR8 Implementation Spartan-II, Spartan-IIE, Virtex, Virtex-E
XOR8 Implementation Spartan-3, Virtex-II, Virtex-II Pro, Virtex-II Pro X

XOR9 Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, these design elements can be inferred or instantiated.

VHDL Inference Code

```vhdl
architecture Behavioral of xor2 is
begin
  process (I0, I1)
  begin
    O <= I0 xor I1;
    end process;

end Behavioral;
```

Verilog Inference Code

```verilog
always @ (I0 or I1)
begin
  O <= I0^I1;
end
```
VHDL Instantiation Template for XOR5

Following is the VHDL code for XOR5. To instantiate XOR2, remove I2, I3, and I4. To instantiate XOR3, remove I3 and I4. For XOR4, remove I4. XOR2B1, and XOR2B2 have the same code as XOR2. XOR3B1, 3B2, and 3B3 have the same code as XOR3 and so forth.

-- Component Declaration for XOR5 should be placed
-- after architecture statement but before begin keyword

component XOR5
    port (O : out STD_ULOGIC;
         I0 : in STD_ULOGIC;
         I1 : in STD_ULOGIC;
         I2 : in STD_ULOGIC;
         I3 : in STD_ULOGIC;
         I4 : in STD_ULOGIC);
end component;

-- Component Attribute specification for XOR5
-- should be placed after architecture declaration but
-- before the begin keyword

-- Attributes should be placed here

-- Component Instantiation for XOR5 should be placed
-- in architecture after the begin keyword

XOR5_INSTANCE_NAME : XOR5
    port map (O => user_O,
              I0 => user_I0,
              I1 => user_I1,
              I2 => user_I2,
              I3 => user_I3,
              I4 => user_I4);

Verilog Instantiation Template for XOR5

XOR5 XOR5_instance_name (.O (user_O),
                         .I0 (user_I0),
                         .I1 (user_I1),
                         .I2 (user_I2),
                         .I3 (user_I3),
                         .I4 (user_I4));

Commonly Used Constraints

None
XORCY

XOR for Carry Logic with General Output

Architectures Supported

<table>
<thead>
<tr>
<th>Architecture</th>
<th>XORCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

XORCY is a special XOR with general O output used for generating faster and smaller arithmetic functions.

Its O output is a general interconnect. See also “XORCY_D” and “XORCY_L”.

Commonly Used Constraints

BEL
XORCY_D

XOR for Carry Logic with Dual Output

Architectures Supported

<table>
<thead>
<tr>
<th>XORCY_D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIe</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

XORCY_D is a special XOR used for generating faster and smaller arithmetic functions.

XORCY_D has two functionally identical outputs, O and LO. The O output is a general interconnect. The LO output is used to connect to another output within the same CLB slice.

See also “XORCY” and “XORCY_L.”

Commonly Used Constraints

BEL
XORCY_L

XOR for Carry Logic with Local Output

Architectures Supported

<table>
<thead>
<tr>
<th>XORCY_L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II, Spartan-IIIE</td>
<td>Primitive</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex, Virtex-E</td>
<td>Primitive</td>
</tr>
<tr>
<td>Virtex-II, Virtex-II Pro, Virtex-II Pro X</td>
<td>Primitive</td>
</tr>
<tr>
<td>XC9500, XC9500XV, XC9500XL</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner XPLA3</td>
<td>No</td>
</tr>
<tr>
<td>CoolRunner-II</td>
<td>No</td>
</tr>
</tbody>
</table>

XORCY_L is a special XOR with local LO output used for generating faster and smaller arithmetic functions. The LO output is used to connect to another output within the same CLB slice.

See also “XORCY” and “XORCY_D.”

Commonly Used Constraints

BEL