Logic Implementation on a Xilinx FPGA using VHDL
WWU Linux platform assumed.
rev 10/28/18

The following is a general outline of steps (i.e. design flow) used to implement a digital system described with VHDL on a Xilinx FPGA.

- Create a directory for your design in which the VHDL and other files for this design will be placed.
- Start the Xilinx ISE tool suit and create a "project". Specify the folder in which the project is to be created and the name of the project (a folder with the project name will be created.) There will be a box in which the type of top level source file is specified. Select HDL as the top design level.
- Select New Source on the left tool bar to create a source file (hover over icons to find the right one). Specify VHDL Module as the type. This file will be created in your project's directory. A window will open and allow you to specify a list of signal names and their direction which will be placed in the entity port statement (you don't have to specify I/O signals here, you can just type them in with the editor but entering them will allow ISE to create a complete entity). Note: the editor in ISE can be used to edit VHDL source files or any programming type text editor (not a word processor).
- After creating the VHDL source file synthesize and implement the deisgn.
- Create the bitmap file..
- Download the bitmap file to the FPGA using Impact. Test your design.

These notes assume that the reader has done schematic oriented design using the Xilinx ISE software.

VHDL source file details

The VHDL file (or files) must be in "plain text" format that conforms to VHDL syntax standards. If your design has multiple VHDL files, one must be the "top level" module and all external inputs/outputs, i.e. connections to the FPGA, must be made to signals declared in the entity port statement of that top level VHDL file.

There are two ways to specify the pin numbers associated with each signal named in the entity port statement. Either they are declared in the VHDL source file using attribute assignments or they are defined in a .ucf constraints file.

On the next page is shown an example file that creates an XOR gate with the FPGA pin connections defined using attribute statements. Note that the first attribute statement essentially defines a data type for the others. Also note that the pin numbers shown here may not be the correct pin numbers for the FPGA board you are using, so change them as needed (An x3cs1200e part in a ft256 package, i.e. it has 256 pins, is assumed here).

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity myXor is
    port(x,y : in std_logic;
            h : out std_logic);
    attribute LOC : string;
    attribute LOC of x : signal is "R16";    -- sw0
    attribute LOC of y : signal is "R15";    -- sw1
    attribute LOC of h : signal is "H14";    -- led0
End myXor;

Architecture myXorBehav Of myXor is
Begin
    h <= x  xor  y;
End myXorBehav;
```

If an input or output signal is a bus, i.e. a vector, rather than having a separate LOC statement for each individual signal in the bus you will just have one LOC statement for the vector but it will have a list of pin numbers. For example, suppose the simple circuit above were described using a 2-bit bus that brings in the connection from two switches. The description might look like this:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity myXor is
    port(x : in std_logic_vector(1 downto 0);
            h : out std_logic);
    attribute LOC : string;
    attribute LOC of x : signal is "R16, R15";    -- sw1 and sw0, in that order
    attribute LOC of h : signal is "H14";    -- led0
End myXor;

Architecture myXorBehav Of myXor is
Begin
    h <= x(1)  xor  x(0);
End myXorBehav;
```

A second way to define I/O pins is to put the pin number definitions in a user constraints file (.ucf) that is located in the project directory (i.e. folder) and not place attribute LOC statements in the VHDL source file. The constraints file is an ASCII text file. It can be created using any text editor (or using the Xilinx PACE constraints editor, but that is not discussed here). On the class webpage are four constraint files that define the possible inputs and outputs for a WWU FPGA2 board.

These files are:

wwu_fpga2_single.ucf	individual signal names
wwu_fpga2_single_null.ucf	individual signal names, all commented out.
wwu_fpga2_vector.ucf	vectorized signal names
wwu_fpga2_vector_null.ucf.	vectorized signal names, all commented out

Download one of these file to the project folder for your design if you wish to use it.

When the synthesizer reads a constraint file it checks to see that all signals listed in the port statement of the top level entity match one-to-one with names in the constraint file and that there are no extra names in the constraint file. Thus any unused names in the constraints file must be commented out.

Synthesis using the Xilinx software; Creating a new project

0) Create a directory for your new project. .

1) Start the Xilinx ISE version 14.7 software by entering ***ise*** (lower case) in a terminal window or from the graphical user interface under "other" select Xilinx ISE.
   Note: ISE may start up with a prior design loaded. If so, click **File** on the Xilinx tool bar and select **close**.

2) Create a new project by clicking on **File** and selecting **New Project**.
   a) To set the Project Location, use the button with three dots to open the Find Directory pop-up window and navigate to the directory you created for the project.
   b) Also, enter a project name
   c) Select HDL as the type of top-level source for the project.
   d) Click **Next**
   e) The *Project Settings* screen should appear. Set values as follows:

Development Board	None Specified
Product Category	General Purpose
Family	Spartan3E
Device	XC3S1200E
Package	FT256
Speed	-5
Top-level Source Type	HDL (this field may be grayed out)
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Isim (VHDL/Verilog)
Preferred Language	VHDL
VHDL Standard	VHDL-93

   f) Click **Next**
   g) The *project summary* screen will be displayed. Review the info and if there is a mistake you can back up to a prior screen and make corrections. Click **Finish** when ok.
   h) At this point there are no design files for the project. In the left window Empty View will be displayed.

i) On the left tool bar at the top is a **New Source button**. Click on it. The *New Source Wizard* will open. Click **VHDL Module** (it should become highlighted) and enter a file name that describes your design. Click **Next**.

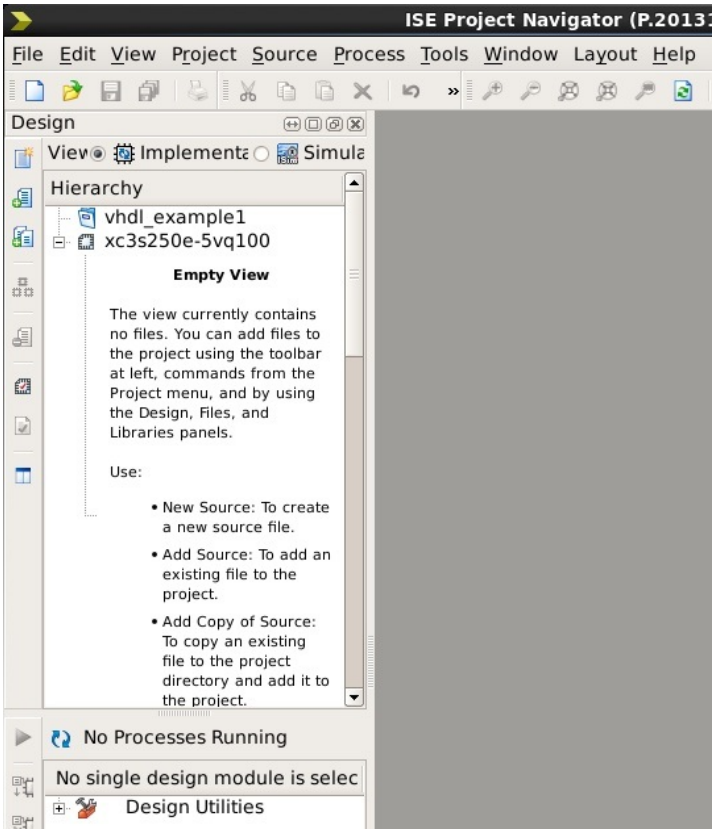
The Define Module window will open and allow you to specify the signals that will connect to physical pins of the FPGA, both input and output. These become the signals named in the port statement in the entity. After listing your signals click **Next.**

The Summary window will open showing the definition of the module and the port definitions. If it isn't ok click back to a prior window. When all is ok click **Finish**. Your newly created VHDL source file should open in the editor window.

Note:
If you created your VHDL source file with another editor you could use the Add Source button on the left tool bar rather than New Source.

j) The Entity of your VHDL description has been created and the start of the Architecture. Two things need to be done before you build the project.
1) Add VHDL statements to the architecture to describe the desired circuit
2) Add attribute statements to the entity to map I/O signals to physical FPGA pins or set up a user constraints file (UCF).
The example circuit below has two input signals that are ANDed together with the result displayed on Led0. Note the attribute statements that define connections to the FPGA

3) The content of the left subwindow(s) changes depending on the tab selected at the bottom.



The default tab is Design which causes a hierarchy display with the name of your project, the FPGA type, and under the FPGA the name of the top level VHDL file.

From this point on building the project and downloading it to an FPGA follows the procedure used for schematic driven design.

Notes:

A) The ibuf and obuf components that are manually placed in a schematic based design are automatically inserted by the synthesizer in a VHDL design based on the signal type declared in the entity port statement.

B) Global clock buffers will automatically be placed on the inputs that are used for global clock inputs. If an internally derived clock needs to be connected to a global clock buffer that will need to be done manually in the VHDL source file like this:

```
Library UNISIM;
use UNISIM.vcomponents.all;

BUFG_inst: BUFG
        port map (
                0 => signal_out;        - - Clock buffer output
                I => signal_in          - - Clock buffer input
        );
```

where signal_out and signal_in are the names of signals you are using in your design.

Initial VHDL file contents:

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    13:20:29 10/27/2015
6  -- Design Name:
7  -- Module Name:    ex1_top_vhdl_module - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27
28 entity ex1_top_vhdl_module is
29     Port ( x : in  STD_LOGIC;
30            y : in  STD_LOGIC;
31            out1 : out  STD_LOGIC);
32 end ex1_top_vhdl_module;
33
34 architecture Behavioral of ex1_top_vhdl_module is
35
36 begin
37
38
39 end Behavioral;
40
```

Main part of the VHDL file with a circuit installed and I/O connections made:

```
28  entity ex1_top_vhdl_module is
29      Port ( x : in  STD_LOGIC;
30             y : in  STD_LOGIC;
31             out1 : out  STD_LOGIC);
32      attribute LOC: string;
33      attribute LOC of x: signal is "P15";  -- sw8
34      attribute LOC of y: signal is "P22";  -- sw9
35      attribute LOC of out1: signal is "P32"; -- led0
36  end ex1_top_vhdl_module;
37
38  architecture Behavioral of ex1_top_vhdl_module is
39
40  begin
41     out1 <= x and y;
42  end Behavioral;
```