Engr433 Xilinx VHDL Capture Tutorial

Revised 10/24/2019

1) Create a new project directory.
2) Start the Xilinx ISE version 14.7 software by entering ***ise*** (lower case) in a terminal window. If you are using a graphical user interface, select **other**, then **Xilinx ISE**. Note that *ise* may start up with a prior design loaded. If so, click **File → Close Project**.
3) Create a new project by clicking on the **New Project…** button in the left-side menu.
   a. To set the Project Location, use the button in the Location line with three dots to open the **Select Directory** pop-up window and navigate to the directory you created for the project.
   b. For the purposes of this handout, enter a project name of **ex1_top**.
   c. Select **HDL** as the type of top-level source and then click **Next.**
   d. The *Project Settings* screen should appear. Set values as follows:

Evaluation Dev Board	None Specified
Product Category	General Purpose
Family	Spartan6
Device	XC6SLX16
Package	FTG256
Speed	-3
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

   e. Click **Next.**
   f. The *project summary* screen will be displayed. Review the information and fix mistakes as needed, then click **Finish**. At this point, there are no design files for the project. In the upper left window, **Empty View** will be displayed.
   g. Click on the **New Source** button in the upper left. The *New Source Wizard* will open. Click on **VHDL Module** and enter the file name *ex1_top_vhdl_module*. 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. For this example, enter:

x	in
y	in
out1	out

   After listing your signals, click **Next.**
   h. The **Summary** window will open showing the definition of the module and the port definitions. Click **Back** to fix anything, otherwise 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 can use the **Add Source** button on the left tool bar rather than **New Source** button.
   i. The **Entity** of your VHDL description has been created as well as the start of the **Architecture** description. Your screes should look something like the listing on the following page.

```
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
```

Figure 1 - Initial VHDL File

4) Two more things need to be done before you build the project:
   i. Add VHDL statements to the architecture to describe the desired circuit.
   ii. 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 pins.

```
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;
```

5) The content of the left sub-window(s) changes depending on the tab selected at the bottom. The default tab is **Design** which causes a hierarchy display in the top left window 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 same procedure as that used for schematic driven design.

Notes:

1. 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.
2. 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, then 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.

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 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.

The next page shows 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 are for the WWU FPGA3 board which uses an XC6SLX16 part in a FTG256 package.

```
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 "P9";  -- sw0
         attribute LOC of y : signal is "N8";  -- sw1
         attribute LOC of h : signal is "L14";  -- 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 can 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 "N8, P9";  -- sw1 and sw0, in that order
         attribute LOC of h : signal is "L14";  -- 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 and not place attribute LOC statements in the VHDL source file. The constraints file is an ASCII text file and 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 two constraint files that define the possible inputs and outputs for a WWU FPGA3 board. These files are:

wwu_fpga3_single.ucf	individual signal names, all commented out
wwu_fpga3_vector.ucf	vectorized signal names, all commented out

Add one of these files 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.