EECE 444 Microprocessor System Design

LAB 1

Part 1: Get familiar with Xilinx 14.7

Download and install Xilinx "ISE Design Suite - 14.7 Full Product Installation "

Use the link below (don't choose the Multi-file download, scroll down to the full product installation):

http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html

If you have a windows 8 machine and the installation doesn't work, please follow the instructions in this link after installation.

http://www.eevblog.com/forum/microcontrollers/guide-getting-xilinx-ise-to-work-with-windows-8-64-bit/

You will need to register, makes sure to get the full free license not the 30 days one. (The download is free after registration).

To start ISE from the start menu

Start ->All Programs ->Xilinx Design Tools -> ISE Design Suit 14.7 -> ISE Design Tools -> Project Navigator

Step 1: create a new project:

1. Select File > New Project... The New Project Wizard appears.

2. Type **project1** in the Project Name field.

3. Enter or browse to a location (directory path) for the new project. A project1 subdirectory is created automatically.

4. Verify that HDL is selected from the Top-Level Source Type list.

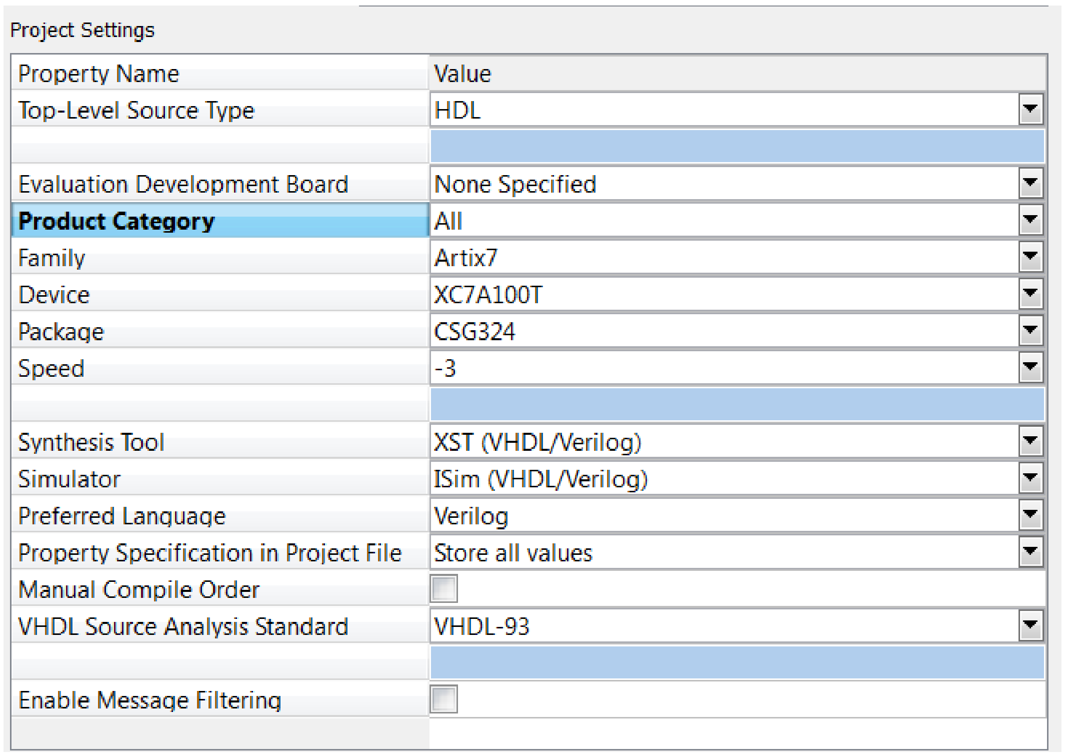
5. Click Next to move to the device properties page.

6. Fill in the properties in the table as shown below:

- Evaluation Development Board: None Specified
- Product Category: General Purpose
- Family: Artix7
- Device: XC7A100T
- Package: CSG324

- Speed Grade: -3
- Top-Level Source Type: HDL
- Synthesis Tool: XST (VHDL/Verilog)
- Simulator: ISE Simulator (VHDL/Verilog)
- Preferred Language: Verilog
- Leave the default values in the remaining fields.

When the table is complete, your project properties will look as shown in figure 1:



Project Settings

Property Name	Value
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	All
Family	Artix7
Device	XC7A100T
Package	CSG324
Speed	-3
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Figure 1

7. Click next, then finish.

Step 2: Created a Verilog HDL Source file

Create the top-level Verilog source file for the project as follows:

1. Right click on xc7a100t-3csg324 in the hierarchy window and select new source.

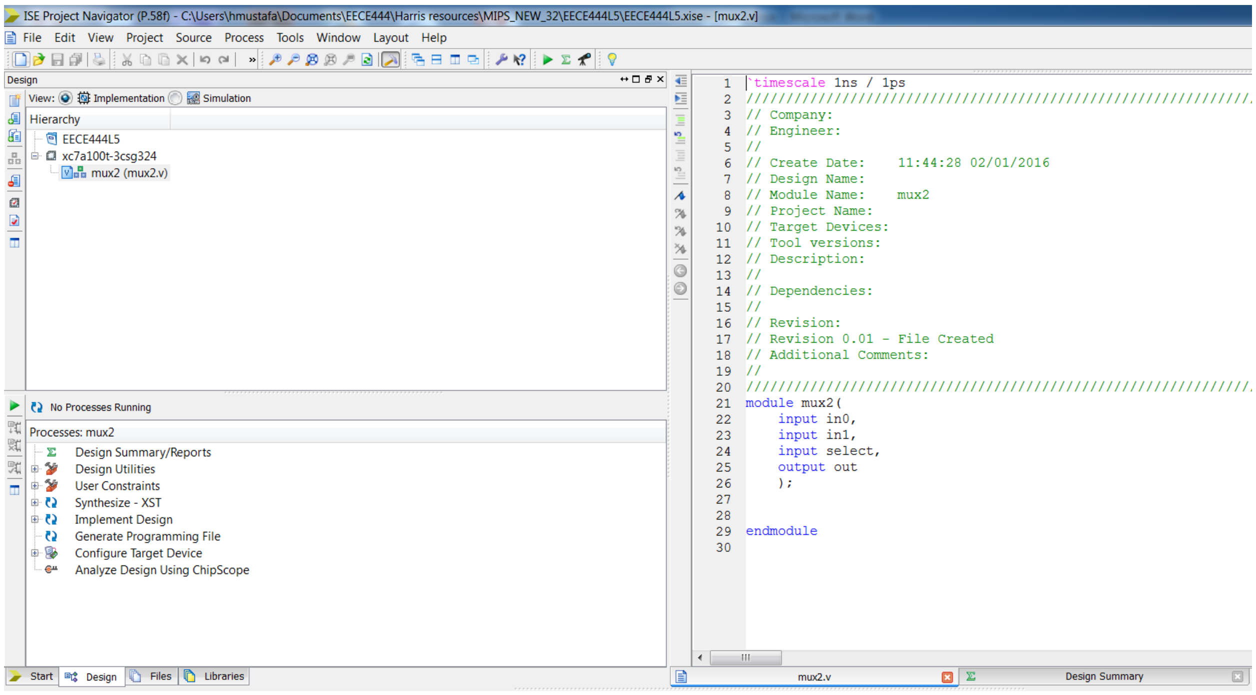
2. Select Verilog Module as the source type in the New Source dialog box.

3. Type in the file name mux2.v.

4. Verify that “Add to Project:” checkbox is selected then click next.

6. Declare the ports for the 2x1 mux design by filling in the port information (in0, in1, select as inputs and out as output)

7. Click Next, then Finish in the New Source Information dialog box to complete the new source file template, as shown in figure 2.


ISE Project Navigator (P.58f) - C:\Users\hmustafa\Documents\EECE444\Harris resources\MIPS_NEW_32\EECE444L5\EECE444L5.xise - [mux2.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
EECE444L5
xc7a100t-3csg324
mux2 (mux2.v)
No Processes Running
Processes: mux2
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries
mux2.v
Design Summary

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    11:44:28 02/01/2016
// Design Name:
// Module Name:    mux2
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module mux2(
    input in0,
    input in1,
    input select,
    output out
    );


endmodule
```

Figure 2

Step 3: Edit Verilog Source

Enter the following Verilog code to describe 2x1 multiplexer

assign out= (in0 & ~select) | (in1 & select);

Step 4: Checking the syntax:

When the source file is complete, you need to check for syntax errors:

1. Verify that Implementation is selected on the design window.

2. Select the mux2.v design source to display the related processes in the Processes window.

3. Click the “+” next to the Synthesize-XST process to expand the process group.

4. Double-click the Check Syntax process.

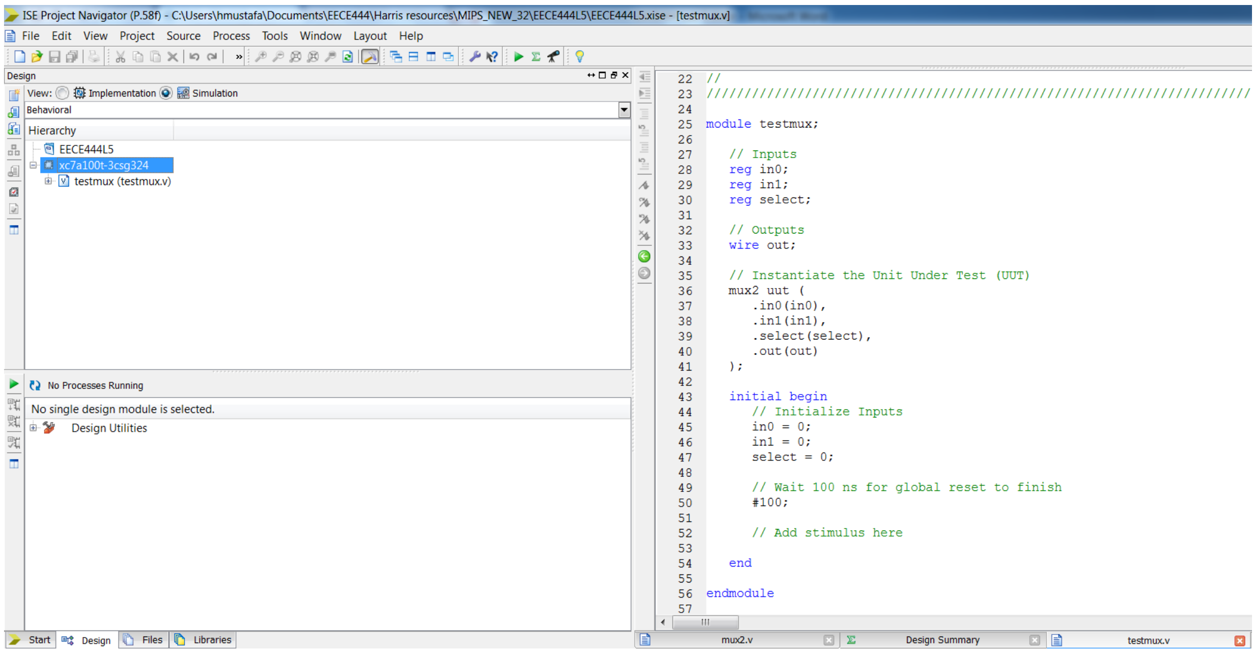
Note: You must correct any errors found in your source files. You can check for errors in the Console tab of the Transcript window. If you continue without valid syntax, you will not be able to simulate or synthesize your design.

5. If your code does not have any errors, you will have a green check mark next to Check Syntax.

Step 5: Simulate your design using ISIM and Verilog testbench

In this step you will be creating a testbench to test your circuit in the simulator.

1. Make sure the ISE mode is changed into “Simulation” form “Implementation” on the Design window
2. Create a new source “Verilog Test Fixture” source and associate it with the only Verilog source you have “mux2.v”. You should have a window similar to figure 3 below after you finish.


ISE Project Navigator (P.58f) - C:\Users\hmustafa\Documents\EECE444\Harris resources\MIPS_NEW_32\EECE444L5\EECE444L5.xise - [testmux.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EECE444L5
xc7a100t-3csg324
testmux (testmux.v)
Design Utilities
Start Design Files Libraries
mux2.v Design Summary testmux.v

No Processes Running

No single design module is selected.

```verilog
//
////////////////////////////////////////////////////////////////////////////////

module testmux;

   // Inputs
   reg in0;
   reg in1;
   reg select;

   // Outputs
   wire out;

   // Instantiate the Unit Under Test (UUT)
   mux2 uut (
      .in0(in0),
      .in1(in1),
      .select(select),
      .out(out)
   );

   initial begin
      // Initialize Inputs
      in0 = 0;
      in1 = 0;
      select = 0;

      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here

   end

endmodule
```

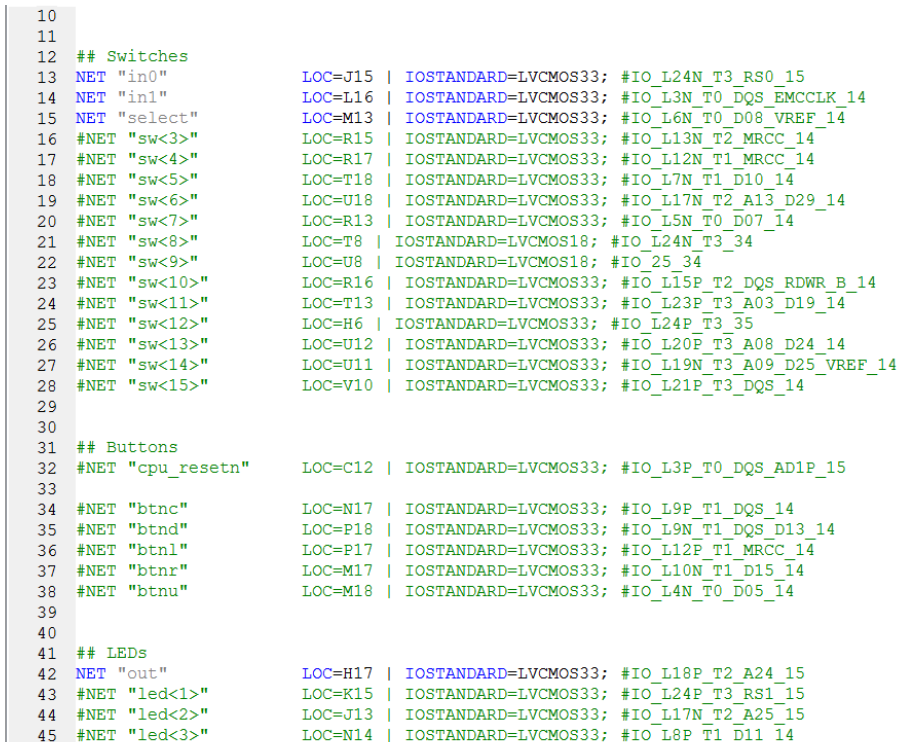
Figure 3

3. Edit the testbench to test for all possible cases. Use $monitor to check for correct answers.
4. Check the schematics and the display result file to ensure all results are correct.

Step 7: Synthesize your design.

Now that you have correctly simulated your design, you will use the ISE tools to synthesize your Verilog code.

1. make sure “Implementation” is selected on the source menu
2. Locate the ***Nexys4DDR_Master.ucf*** file “provided on Blackboard” and add a “copy of a source file” to your project.
3. Open the ucf file and edit the first 3 switches and the first LED line as shown in figure 4.



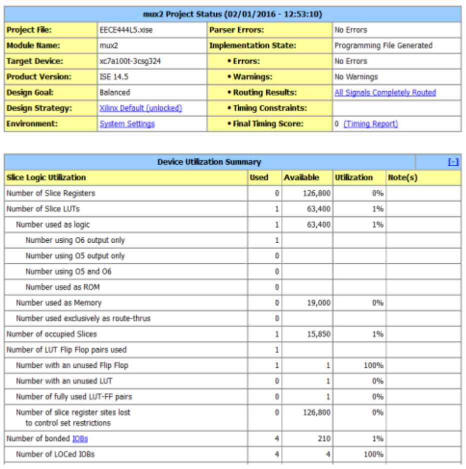
```


## Switches
NET "in0"             LOC=J15 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_RS0_15
NET "in1"             LOC=L16 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_EMCCLK_14
NET "select"          LOC=M13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_D08_VREF_14
#NET "sw<3>"          LOC=R15 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_14
#NET "sw<4>"          LOC=R17 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_14
#NET "sw<5>"          LOC=T18 | IOSTANDARD=LVCMOS33; #IO_L7N_T1_D10_14
#NET "sw<6>"          LOC=U18 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A13_D29_14
#NET "sw<7>"          LOC=R13 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_D07_14
#NET "sw<8>"          LOC=T8  | IOSTANDARD=LVCMOS18; #IO_L24N_T3_34
#NET "sw<9>"          LOC=U8  | IOSTANDARD=LVCMOS18; #IO_25_34
#NET "sw<10>"         LOC=R16 | IOSTANDARD=LVCMOS33; #IO_L15P_T2_DQS_RDWR_B_14
#NET "sw<11>"         LOC=T13 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_A03_D19_14
#NET "sw<12>"         LOC=H6  | IOSTANDARD=LVCMOS33; #IO_L24P_T3_35
#NET "sw<13>"         LOC=U12 | IOSTANDARD=LVCMOS33; #IO_L20P_T3_A08_D24_14
#NET "sw<14>"         LOC=U11 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_A09_D25_VREF_14
#NET "sw<15>"         LOC=V10 | IOSTANDARD=LVCMOS33; #IO_L21P_T3_DQS_14


## Buttons
#NET "cpu_resetn"     LOC=C12 | IOSTANDARD=LVCMOS33; #IO_L3P_T0_DQS_AD1P_15

#NET "btnc"           LOC=N17 | IOSTANDARD=LVCMOS33; #IO_L9P_T1_DQS_14
#NET "btnd"           LOC=P18 | IOSTANDARD=LVCMOS33; #IO_L9N_T1_DQS_D13_14
#NET "btnl"           LOC=P17 | IOSTANDARD=LVCMOS33; #IO_L12P_T1_MRCC_14
#NET "btnr"           LOC=M17 | IOSTANDARD=LVCMOS33; #IO_L10N_T1_D15_14
#NET "btnu"           LOC=M18 | IOSTANDARD=LVCMOS33; #IO_L4N_T0_D05_14


## LEDs
NET "out"             LOC=H17 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A24_15
#NET "led<1>"         LOC=K15 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_RS1_15
#NET "led<2>"         LOC=J13 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A25_15
#NET "led<3>"         LOC=N14 | IOSTANDARD=LVCMOS33; #IO_L8P_T1_D11_14
```

Figure 4

4. Double click on “Implement design” and make sure you receive a green check mark after the process finishes.

Step 6: Implement your Design and generate programming file:

1. Execute Implement Design from the processes window
2. Double click Generate Programming File to generate a .bit file which will be used to program the board.
3. Check the design summary: as the project progresses, a report are generated in each process. All reports can be checked on the design summary window. Also you can look at the RTL schematics by double clicking on “View RTL Schematics”, see Figure 5.



mux2 Project Status (02/01/2016 - 12:53:10)			
Project File:	EECE444L5.xise	**Parser Errors:**	No Errors
Module Name:	mux2	**Implementation State:**	Programming File Generated
Target Device:	xc7a100t-3csg324	• **Errors:**	No Errors
Product Version:	ISE 14.5	• **Warnings:**	No Warnings
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	0	126,800	0%	
Number of Slice LUTs	1	63,400	1%	
Number used as logic	1	63,400	1%	
Number using O6 output only	1			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as ROM	0			
Number used as Memory	0	19,000	0%	
Number used exclusively as route-thrus	0			
Number of occupied Slices	1	15,850	1%	
Number of LUT Flip Flop pairs used	1			
Number with an unused Flip Flop	1	1	100%	
Number with an unused LUT	0	1	0%	
Number of fully used LUT-FF pairs	0	1	0%	
Number of slice register sites lost to control set restrictions	0	126,800	0%	
Number of bonded IOBs	4	210	1%	
Number of LOCed IOBs	4	4	100%	

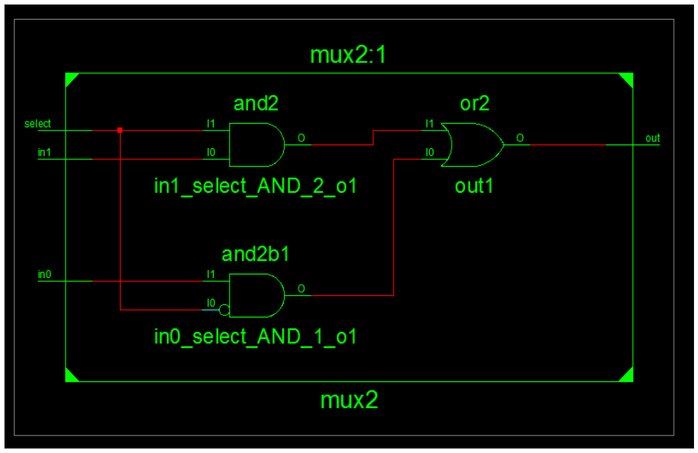
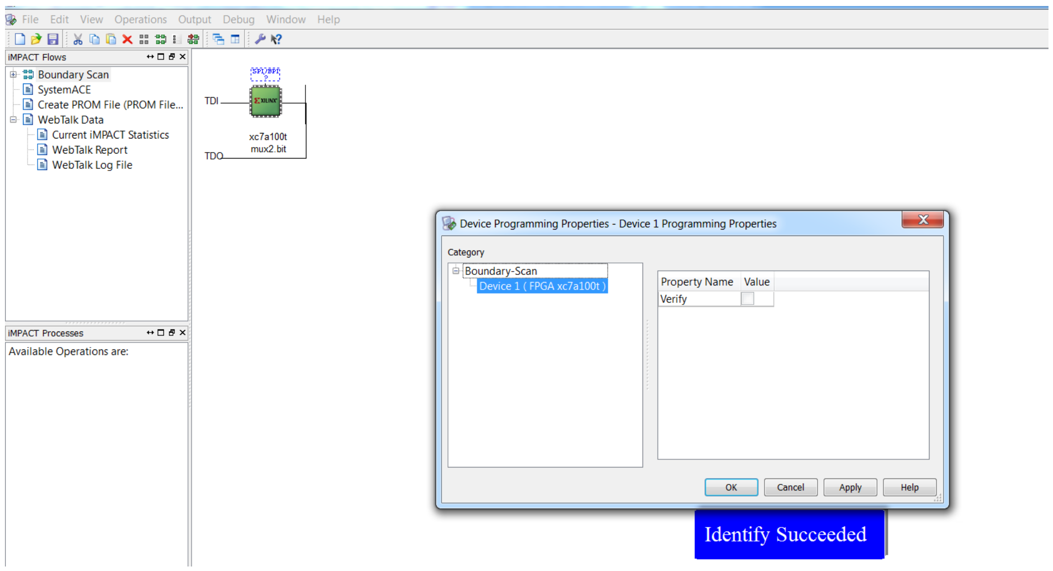


Figure 5

Step 7: Program the board

Before proceeding to the next step connect the board via USB cable to the PC and turn on the power.

1. Double click on Configure Target Device and the ISE will launch another Xilinx tool called iMpact. A warning box will appear complaining about "No iMpact project file exist" click OK
2. In the ISE iMpact window double click **Boundary Scan** in the top left pane. In the boundary scan window in the main pane, where it says "Right click to Add Device or Initialize Chain" right-click in the middle and select **Initialize Chain.** Xilinx will ask you if you want to continue and assign configuration files, click **Yes** and select the "mux2.bit" file from the browsing window that will open. You will be asked if you need to attach an SPI or PROM click **No.**
3. Next you will be asked which device on the board you will program, for our case we only have the FPGA chip, click **Ok** to complete as shown in figure 6.


File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File...
WebTalk Data
Current iMPACT Statistics
WebTalk Report
WebTalk Log File
iMPACT Processes
Available Operations are:
TDI
TDO
xc7a100t
mux2.bit
Device Programming Properties - Device 1 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc7a100t)
Property Name
Value
Verify
OK
Cancel
Apply
Help

Identify Succeeded

Figure 6

4. Right-click on the green chip and select program.
5. Test your circuit for functionality. Note, you don't need to restart your board, if you do you will lose the program and will need to program again.

Part 2: Create your own Binary to BCD decoder

Design a Gate-level greater-than circuit. The greater than circuit compares two inputs, **a** and **b**, and asserts an output **GT** when **a** is greater than **b**. You need to create a 4-bit greater-than circuit from the bottom up and use only gate-level logical operators. Design the circuit as follows:

1. Derive the truth table for a 2-bit greater than circuit and obtain the logic expression in the sum of product SOP format. Based on the expression, derive the Verilog code using only logical operators.
2. Derive a testbench for the 2-bit greater than circuit. Perform simulation and verify the correctness of your design.
3. Use 4 switches as the inputs and one LED as the output. Synthesize the circuit and download the configuration file to the board. Verify the operation.
4. Derive the Verilog code for a 2-bit equality comparator with two inputs i0 and i1, and one output eq. the eq signal will be asserted if i0 and i1 are equal.
5. Derive a testbench for the comparator. Perform simulation and verify the correctness of your design. You don't need to download this design to the board.
6. Use the 2-bit greater-than circuits and the 2-bit equality comparators and a minimal number of gates to construct a 4-bit greater than circuit. First draw a block diagram and then derive the structural Verilog code according to the diagram.
7. Derive a testbench for the 4-bit greater-than circuit. Perform simulation and verify the correctness of the design
8. Use eight switches as the input and one LED as the output. Synthesize the circuit and download the configuration file to the board. Verify the operation.

Demonstration:

Demonstrate your working design to your TA; make sure to get signed off.

What to Turn In

Include the following elements in your final submission.

Please indicate how many hours you spent on this lab. This will not affect your grade but will be helpful for calibrating the workload for next semester's labs.

1. Your design diagram for all components including the top module.
2. Verilog code.
3. Simulation waveforms of the final 4-bit greater-than circuit.