

ECE 2700 Lab 1

Due at the end of your registered lab session (40 points)

Objectives

To become familiar with Xilinx ISE and learn how to load a simple Verilog design onto the Basys Board.

1 Pre-Lab Preparation

Check out the Basys Board from the ECE store prior to your lab session. You should also read this document in its entirety beforehand. Note that it will be useful to bring along the *Verilog for Digital Design* textbook.

2 Resources

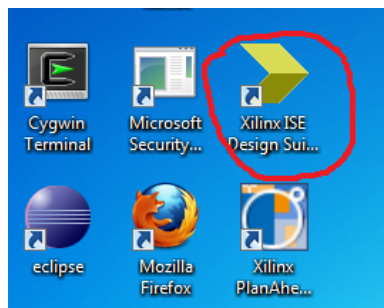
You can download the ISE Webpack (Windows and Linux) and install it on your personal computer. The ISE Webpack should have everything that you will need for the labs this semester.

3 Overview

While the ultimate goal of this lab is to load a simple design onto the Basys board (after you have written and compiled your code), it is always a good idea to simulate your design first to detect any problems right away. Once you are convinced your design works, you will load it onto the board.

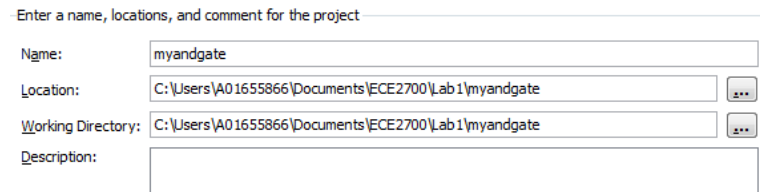
4 Preparation

Inside My Documents, create a directory called ECE2700. Then, create a directory called Lab1 inside your ECE2700 directory. The program we will be using is called Xilinx ISE. To run the program, double click on the Xilinx ISE Design Suite 13.2 icon located on the Desktop, as shown below.



5 Writing Verilog Code and Simulating Your Design

Click on File → New Project. Type in **myandgate** in the Name field. Select the **Lab1** folder for the Location field. Make sure that the Top-level source type is set to HDL, as shown in the following screenshot.



Enter a name, locations, and comment for the project

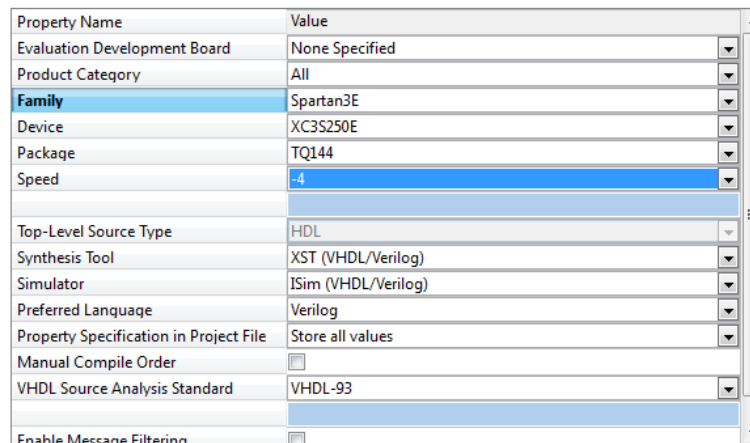
Name: myandgate

Location: C:\Users\A01655866\Documents\ECE2700\Lab1\myandgate

Working Directory: C:\Users\A01655866\Documents\ECE2700\Lab1\myandgate

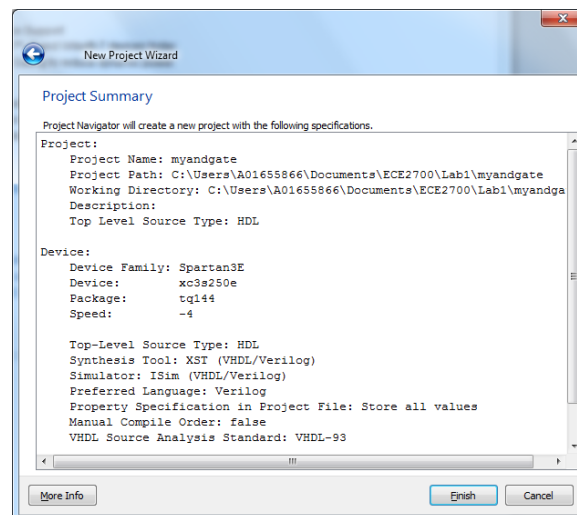
Description:

Click Next. You are now asked to specify some settings for the project. You should specify **Spartan3E** for Family, **XC3S250E** for Device, **TQ144** for Package if you have the original Basys board and **CP132** if you have the Basys 2 board, and **-4** for Speed. **Please carefully examine the chip on your board to make sure that you have specified the correct device.** You may have the **XC3S100E** device instead of the **XC3S250E** device. Click Next.



| Property Name | Value |
|--|--------------------------|
| Evaluation Development Board | None Specified |
| Product Category | All |
| Family | Spartan3E |
| Device | XC3S250E |
| Package | TQ144 |
| Speed | -4 |
| Top-Level Source Type | HDL |
| Synthesis Tool | XST (VHDL/Verilog) |
| Simulator | ISim (VHDL/Verilog) |
| Preferred Language | Verilog |
| Property Specification in Project File | Store all values |
| Manual Compile Order | <input type="checkbox"/> |
| VHDL Source Analysis Standard | VHDL-93 |
| Enable Message Filtering | <input type="checkbox"/> |

You should see something like the following screen. Click Finish.



New Project Wizard

Project Summary

Project Navigator will create a new project with the following specifications.

Project:

- Project Name: myandgate
- Project Path: C:\Users\A01655866\Documents\ECE2700\Lab1\myandgate
- Working Directory: C:\Users\A01655866\Documents\ECE2700\Lab1\myandgate
- Description:
- Top Level Source Type: HDL

Device:

- Device Family: Spartan3E
- Device: xc3s250e
- Package: tq144
- Speed: -4

Top-Level Source Type: HDL

Synthesis Tool: XST (VHDL/Verilog)

Simulator: ISim (VHDL/Verilog)

Preferred Language: Verilog

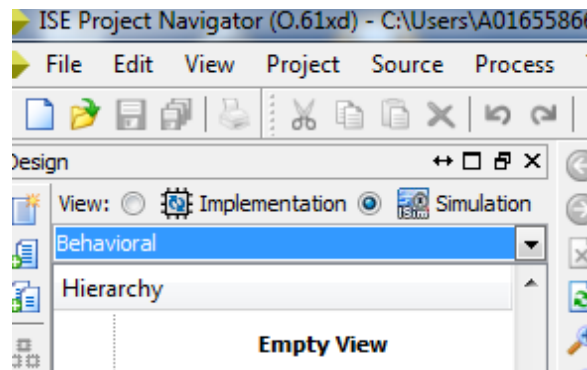
Property Specification in Project File: Store all values

Manual Compile Order: false

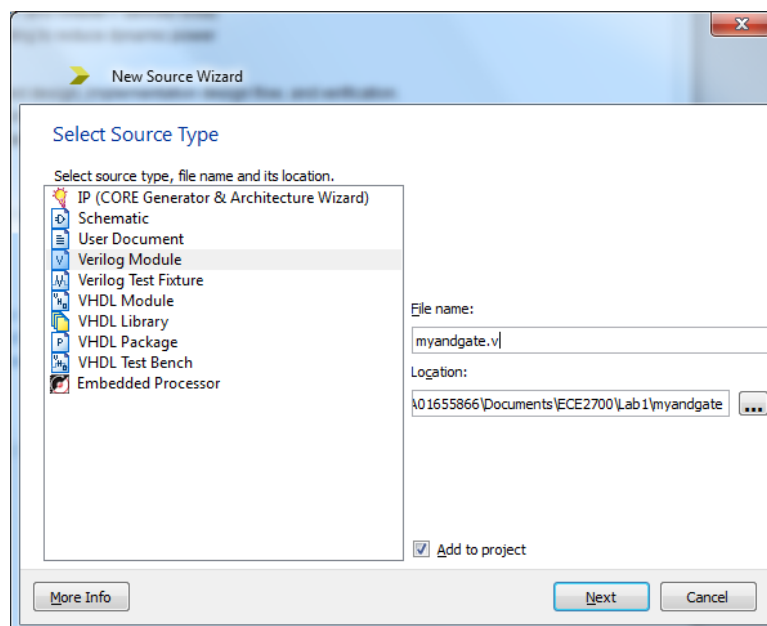
VHDL Source Analysis Standard: VHDL-93

More Info Finish Cancel

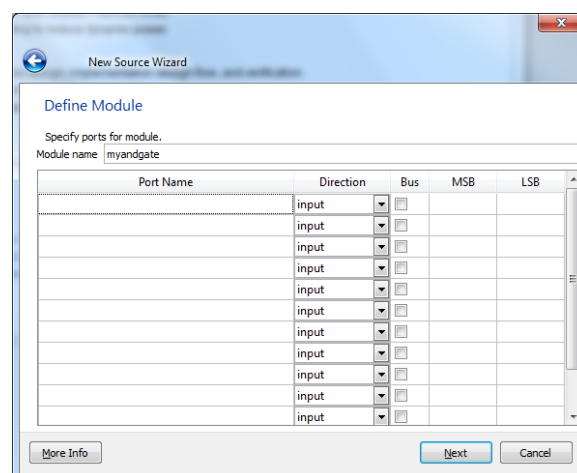
You are now ready to write your first Verilog code. Since we will first simulate our design before implementing it on the Basys board, select the Simulation radio button, as shown below. Make sure the top box is selected to Behavioral.



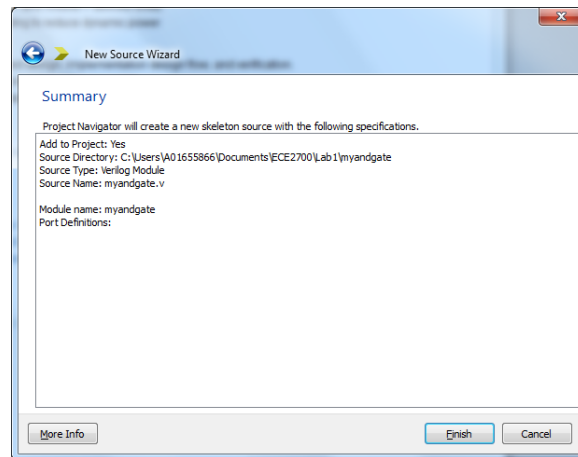
To add a new file to the project, click on Project → New Source. A popup window will open. Select Verilog Module and specify `myandgate.v` as the filename.



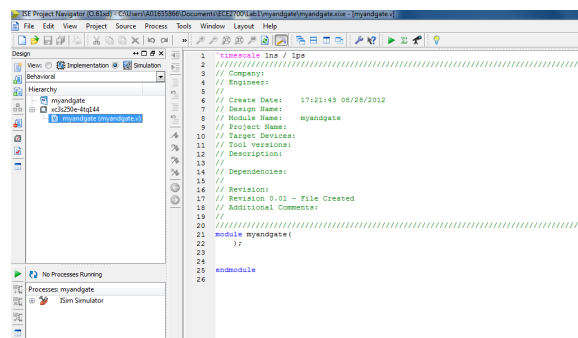
Click on the Next button, and again for the Define Module (you don't have to specify anything here).



A summary window will show up. Click Finish.



Under the Hierarchy window, you should now see your `myandgate` file, as shown below.



Click on it and type in the following code.

```
module myandgate(a, b, F);

    input a, b;
    output F;

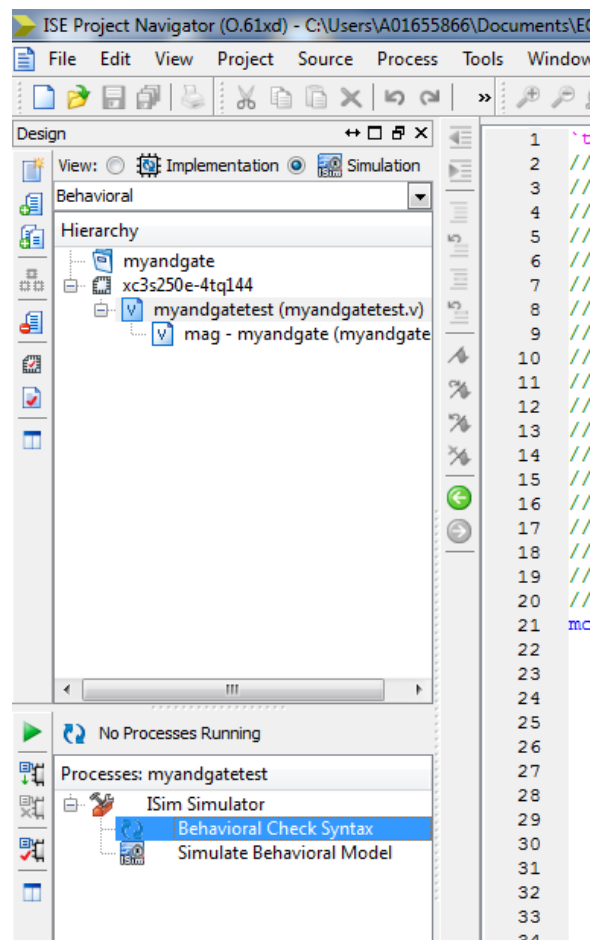
    and and1(F, a, b);

endmodule
```

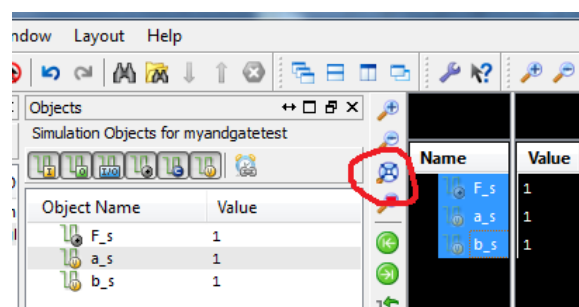
Note that the above piece of code simply uses the built-in AND gate and define a new AND gate in which the input ports appear before the output port.

You will now create a testbench to verify your design. Repeat the above steps for adding a new file. Use `myandgatetest.v` as the filename. Fill in the testbench code with all 4 test cases (see page 16 of *Verilog for Digital Design*). Save your work. Remember that `X_s` and `Y_s` in the testbench correspond to `a` and `b` in your code.

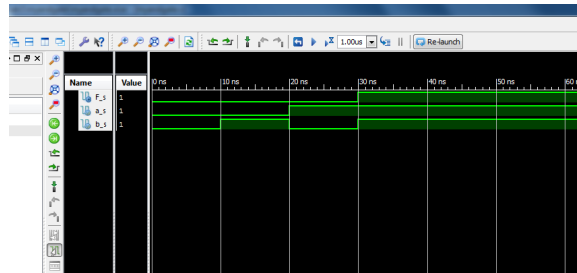
Now we will check for syntax error and simulate your AND gate design. Click on `myandgatetest` in the Hierarchy window. In the Processes window below, expand the ISim Simulator. Double click on Behavioral Check Syntax.



Check the Console to see whether your design contains any syntax error. Fix them if needed then double click on the Simulate Behavioral Model in the Processes window. This should bring up a waveform window. Here, you cannot see the results clearly because of the timescale. Click on the following magnifier icon to see the results.

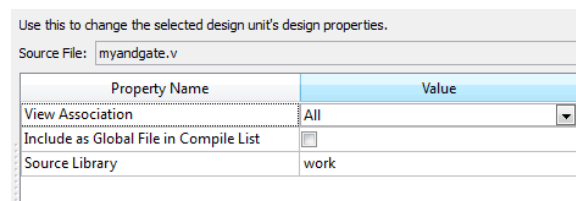


You should see the results for the AND gate similar to the one shown below. Make sure your design is correct

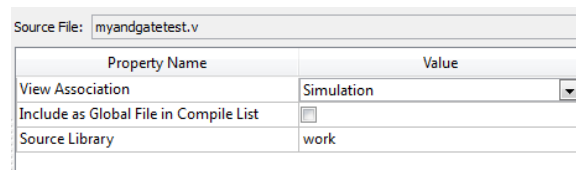


6 Implementation on the Basys Board

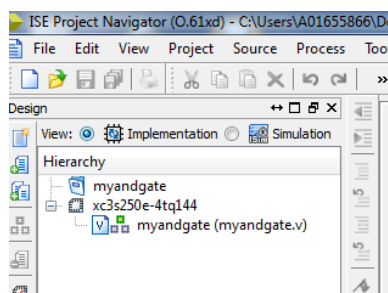
Now that you have verified that your design is correct, you are ready to load your design onto the board. Close the waveform window. In the main window, right click on the `myandgate` Verilog file. Select Source Properties. Under View Association, select All.



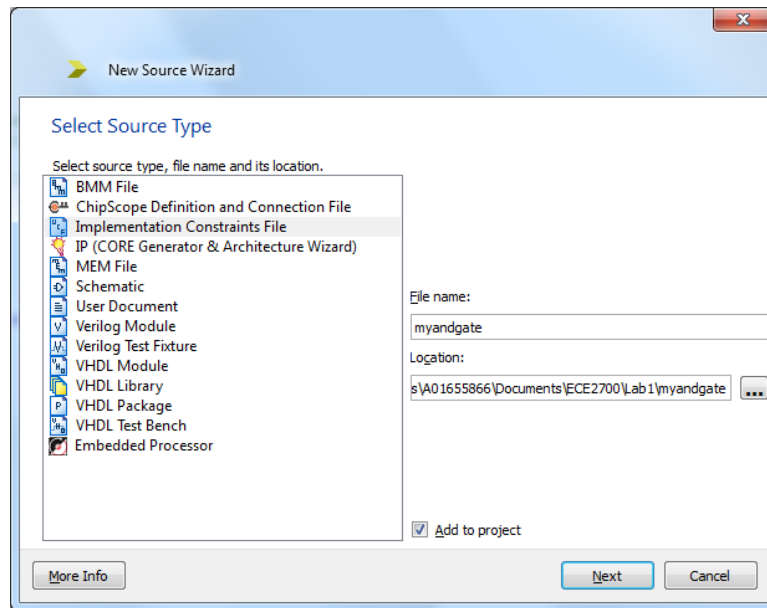
Right click on the `myandgatetest` Verilog file and select Source Properties. Under View Association, select Simulation.



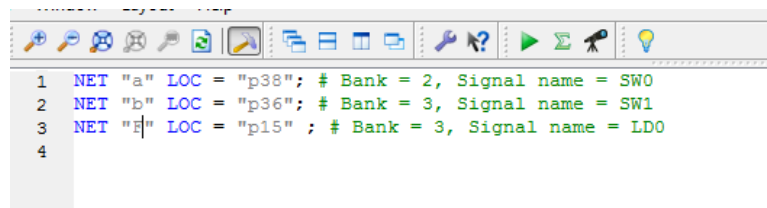
Above the Hierarchy window, select the Implementation radio button.



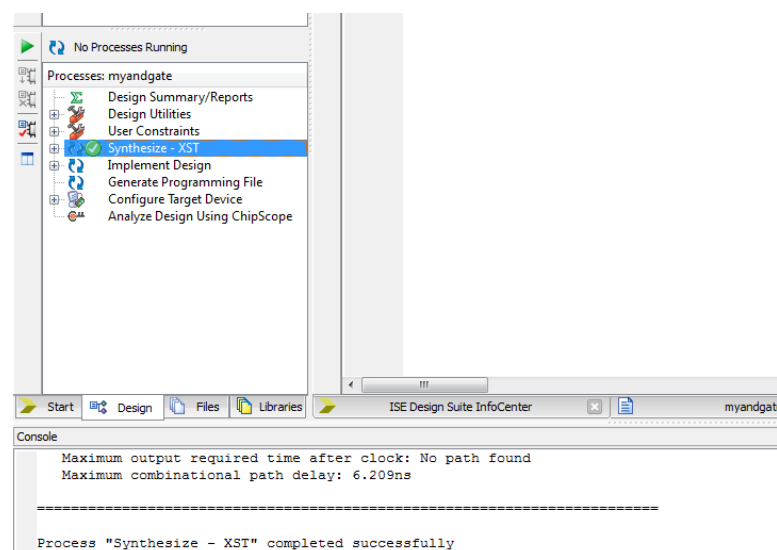
We are almost there. Now we need to tell the program which pins on the board we would like to use. On your Basys board, locate switches `sw0` and `sw1`. Also locate the LED `LD0`. We will map the inputs `a` and `b` to the switches and the output `F` to the LED. The idea is that the LED will light up if both inputs are 1 (thus implementing the AND gate). To assign pins, click on Project → New Source. Select Implementation Constraints File and name it `myandgate`. Click Next then Finish.



Open the UCF file you've just created. Look inside the file `MainBasys.ucf` (or `MainBasys2.ucf` if you are using the Basys 2 board), which can be found on the course website. Locate the lines that contain SW0, SW1, and LD0. Once you find them, copy the 3 lines over to the UCF file in the main window. Change the NET name in each line to **exactly** correspond to your input and output names. **Caution: Do not refer to the little flyer with pin assignment that comes with your kit. Some of the pin numbers are incorrect!**



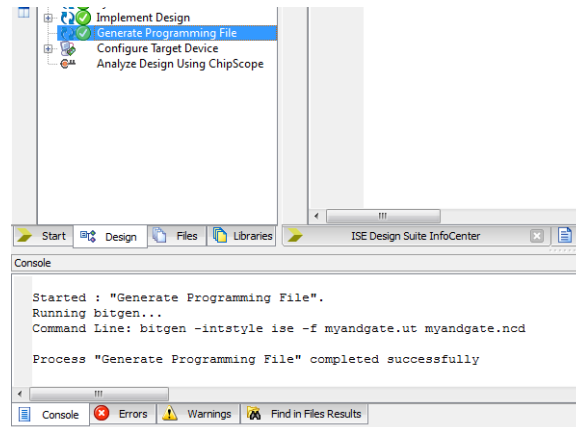
Save your changes. Click on `myandgate` in the Hierarchy window. In the Processes window, double-click on Synthesize - XST. You should see a message indicating that the process was completed successfully.



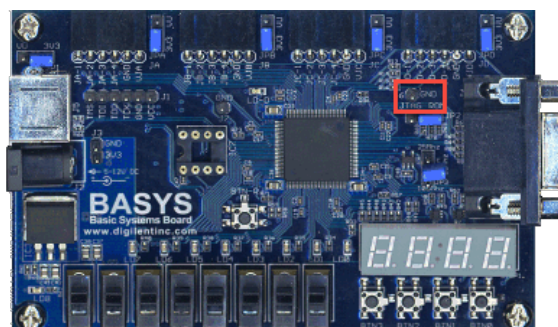
Now you will generate the bit file for the FPGA. To do this, right click on the Generate Programming File option in the Processes window. Select Process Properties then Startup Options. For the FPGA Start-Up Clock, select JTAG Clock. This step ensures that you use the clock on the FPGA. Click OK.

| Switch Name | Property Name | Value |
|----------------|---------------------------------|--------------------------|
| -g StartUpClk: | FPGA Start-Up Clock | JTAG Clock |
| -g DonePipe: | Enable Internal Done Pipe | <input type="checkbox"/> |
| -g DONE_cycle: | Done (Output Events) | Default (4) |
| -g GTS_cycle: | Enable Outputs (Output Events) | Default (5) |
| -g GTS_cycle: | Release Outputs (Output Events) | Default (5) |

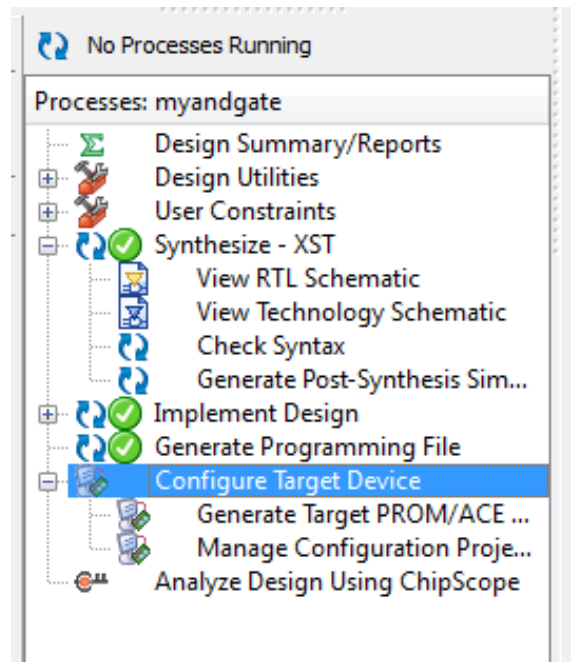
Double click on the Generate Programming File option. This may take a while. Once the process is finished, you should see the following message.



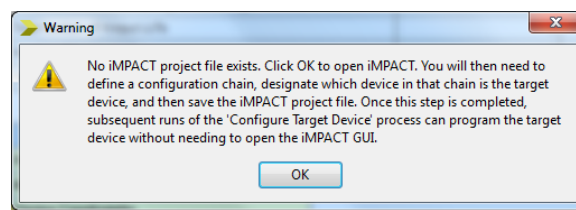
You are finally ready to load your program onto the FPGA. It's time to connect the board to your computer. A red light should illuminate to indicate that a power connection has been established. Make sure SW8 is down to indicate that you will power the board via the USB port for the original Basys board (up for the Basys 2 board). **Very Important! For the original Basys board, make sure the jumper is set to JTAG, not PROM. See the location of this below. For the Basys 2 board, make sure the jumper for the JP3 signal is set to PC instead of ROM.**



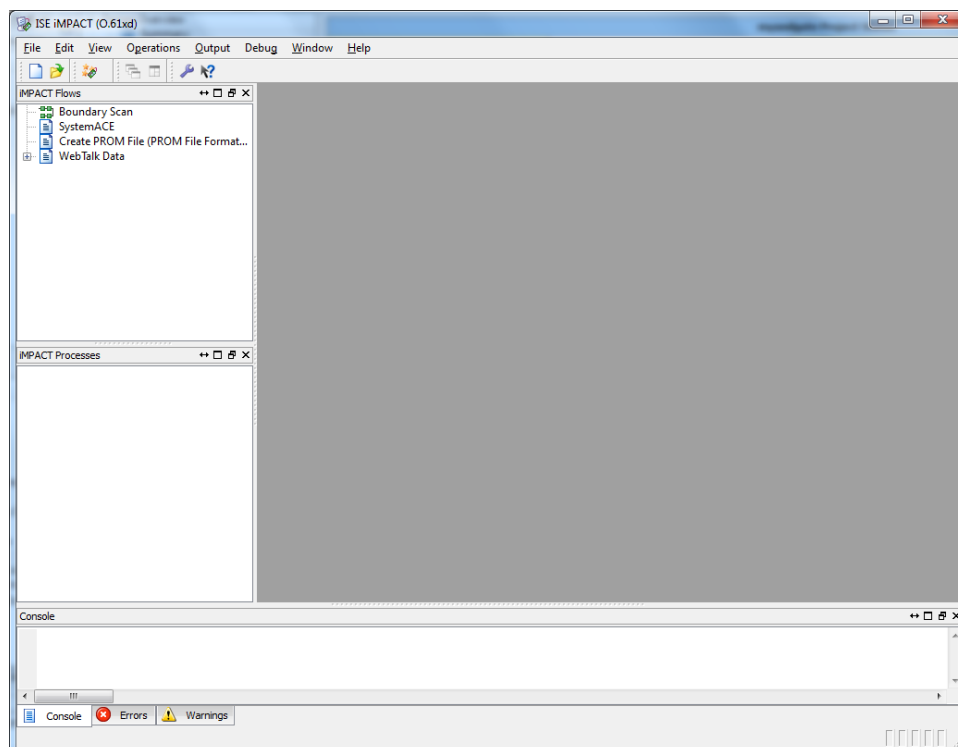
Double click on the Configure Target Device, as shown below.



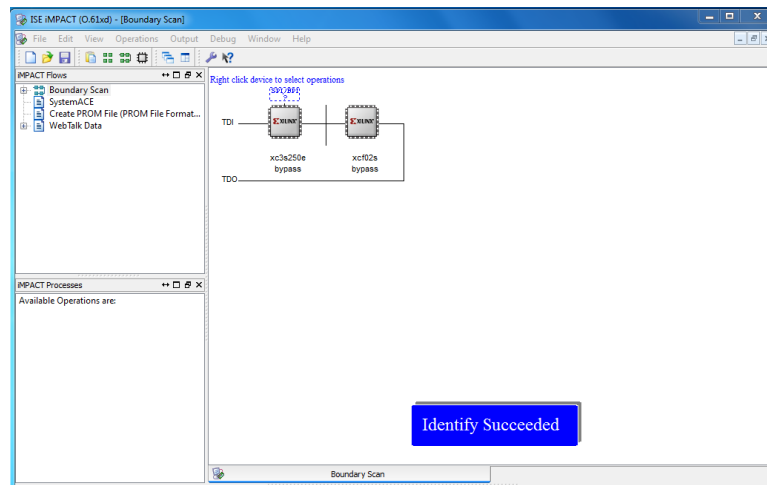
You will see the following warning the first time you do this. Click OK.



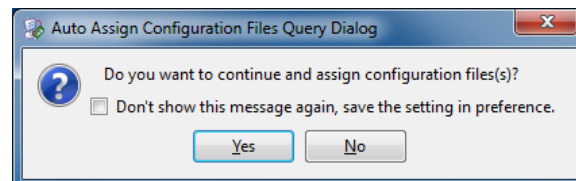
This will bring up a new window. Double click on Boundary Scan inside the left menu bar.



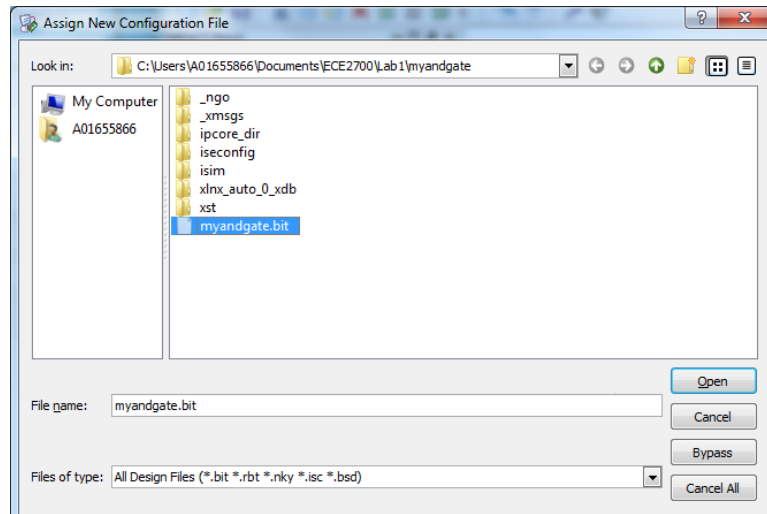
Now, in the main window, right click and select Initialize Chain. You should see the following window shortly after.



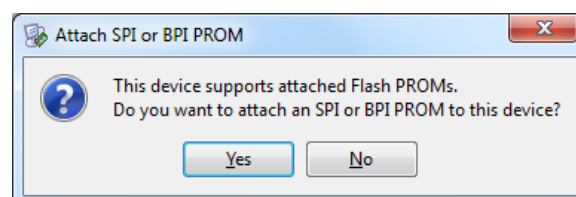
A pop-up window will appear. Click Yes.



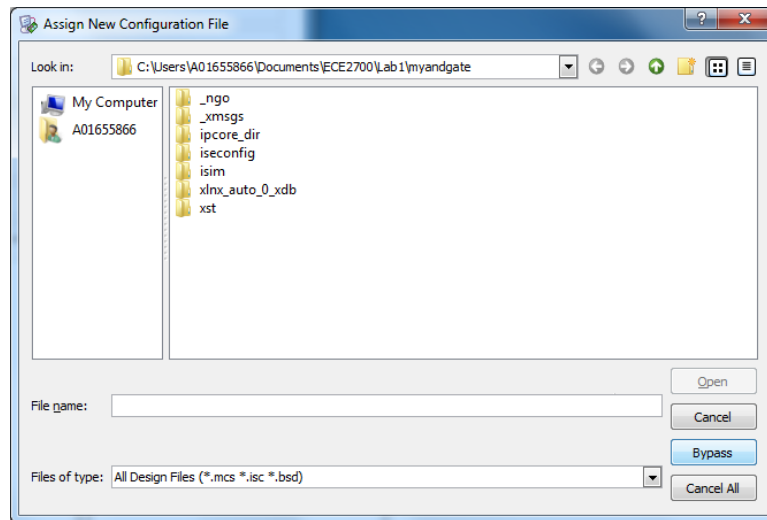
Now select the bit file you have just generated. It should be inside your `myandgate` directory.



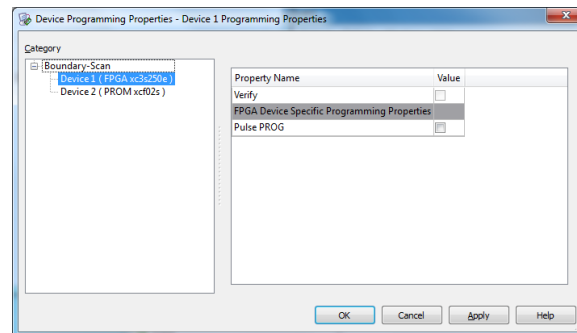
You will see the following pop-up window. Click No.



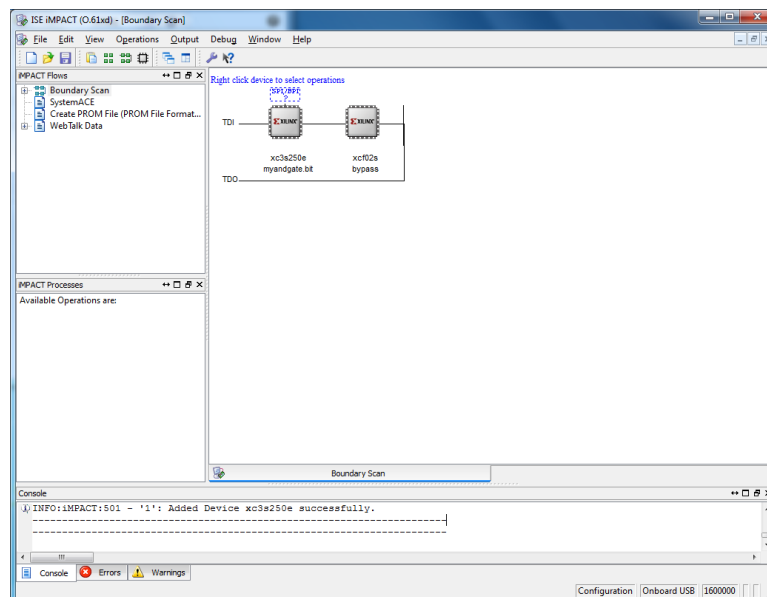
Yet another pop-up window will appear. Click on Bypass.

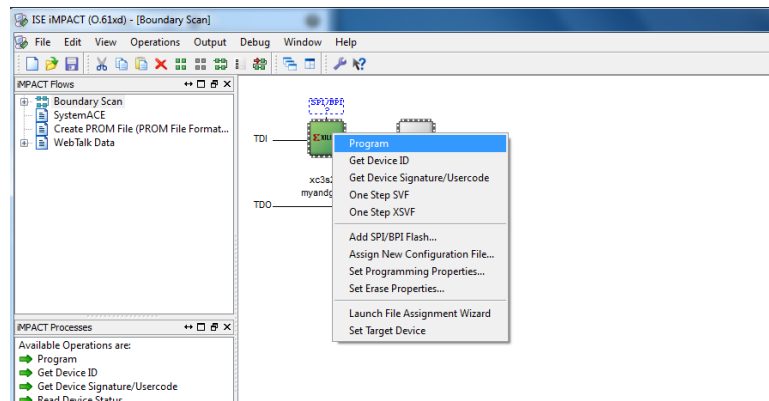


You should now see a summary window. Click OK.

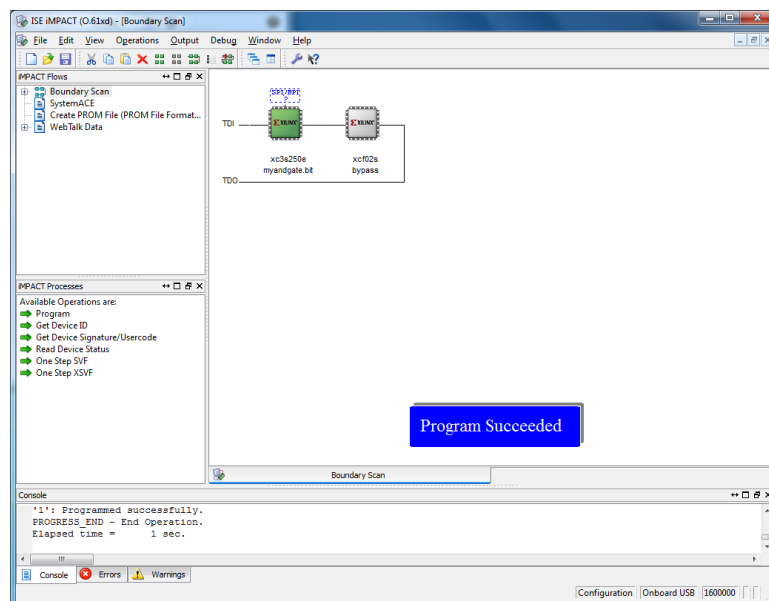


Back to the iMPACT window, the bit file you selected should now be associated with the FPGA. Right click and select Program to load the bit file onto the FPGA.



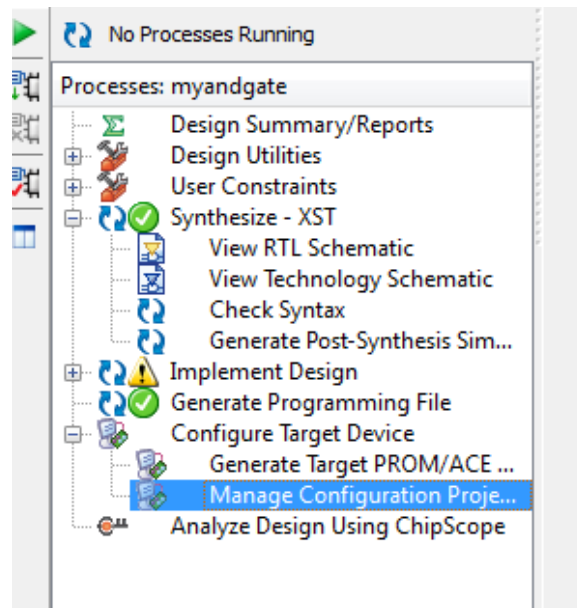


The process should be a success, as shown before.



You have just loaded your design onto the FPGA board. Play with SW0 and SW1. What happens if you flip them up at the same time? That's right, you're looking at your newly built AND gate!

If you make any modification to the Verilog code, be sure to regenerate the bit file. Afterwards, you will need to reload it onto the FPGA. To do this, restart the Manage Configuration Project option as shown below. Then, proceed as before.

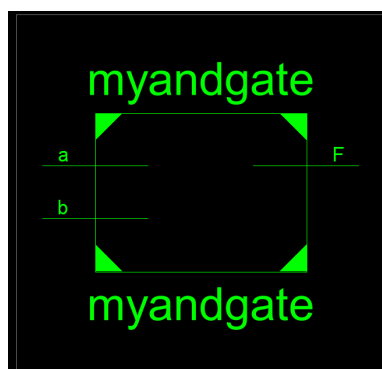


There are many steps for loading your design onto the FPGA board. However, you will become more familiarized with them as the semester progresses. In summary, you need to do the following.

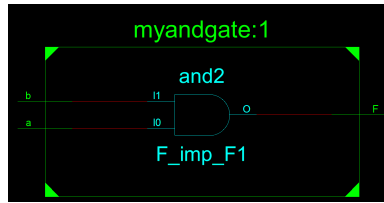
1. Write your Verilog code.
2. Simulate your design to make sure it is correct.
3. Create and generate a UCF file to specify pin assignments.
4. Generate the bit file.
5. Use iMPACT to upload your bit file to the FPGA.

7 Fun Stuff

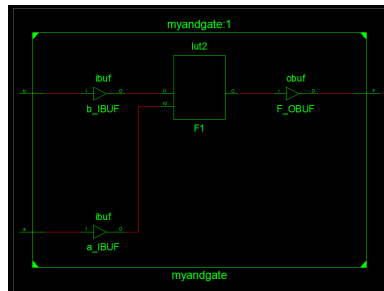
Xilinx ISE has various features and though we will not have time to learn them all, I encourage you to experiment. For example, if you double click on View RTL Scheme (expand the Synthesize - XST option to see it) and click on Start with a schematic of the top-level block, you will see a circuit diagram for your AND gate.



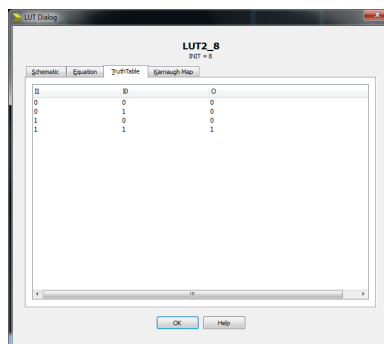
Double clicking on it will show how the design is implemented.



You can also right click on the View Technology Schematic option. Double click on the design will yield the following.



Double clicking inside the LUT (lookup table) box will cause a new window to pop up. You can see the schematic, equation, truth table, and even k-map of your design.



Have fun!

8 TA Checkoff

- (4 points) Complete pre-lab work prior to start of the lab.
- (16 points) Correct simulation of the AND gate.
- (20 points) Correct implementation of the AND gate on the Basys board.

Important: Please upload your .v files and .ucf file on Canvas. Failure to do so will result in a zero for this assignment.