Lab 5

Lab 5 assignment is due on Monday, March 23, by 9pm in Blackboard.

Please ask all questions about this assignment during the office hours or post questions on piazza.

This lab requires installing new software on your Windows laptop. Note that the software is very demanding in terms of computer resources and may not work well on your laptop. **Also, the software is not available for MacOS platform.**

Introduction to Altera Quartus software

The goal of this lab is to introduce you to industry design and simulation tools that can help you design, test, and debug your circuits quickly and efficiently. This is the same software you will be using in ECE 385.

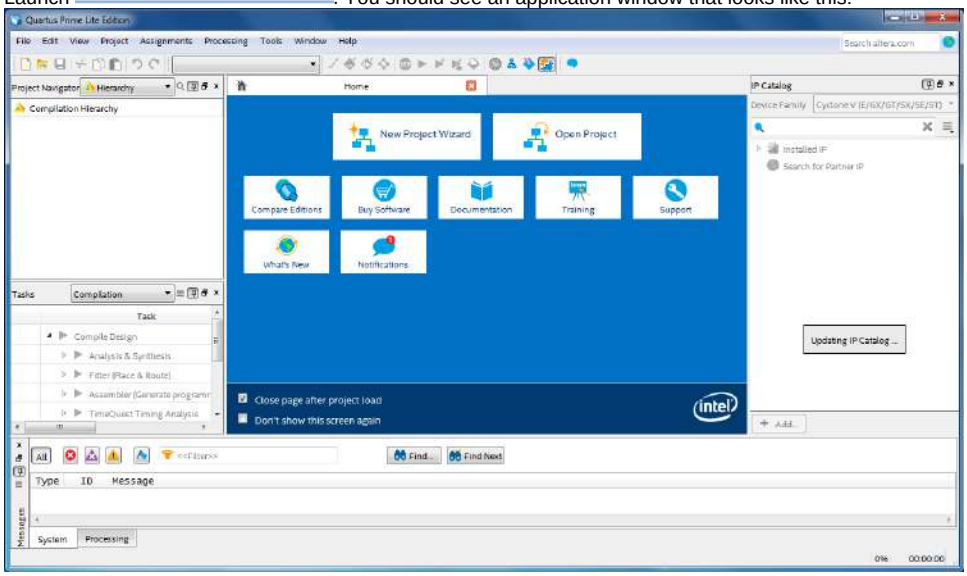
The software is already installed on the workstations in the computer lab D211. But since we do not have remote access to the lab, you must download and install this software on your own Windows laptop. Note that the software is very demanding in terms of computer resources and may not work well on your laptop. **Also, the software is not available for MacOS platform.**

To install the software on your own laptop, start by downloading latest version of the **Quartus Prime Lite edition**, **ModelSim Started editions**, and **Cyclone V device support files** and follow the installation procedure. These tools are free for academic use, but they require a registration with Altera /Intel. First, install Quartus Prime, and then install ModelSim. Take a note of the path where ModelSim was installed, we will need this info later. Both Quatrus Prime and ModelSim must be installed as well as support for at least one device. Strictly speaking, we do not need any devices in this course, however support package for at least one device is needed for the tools to function properly.

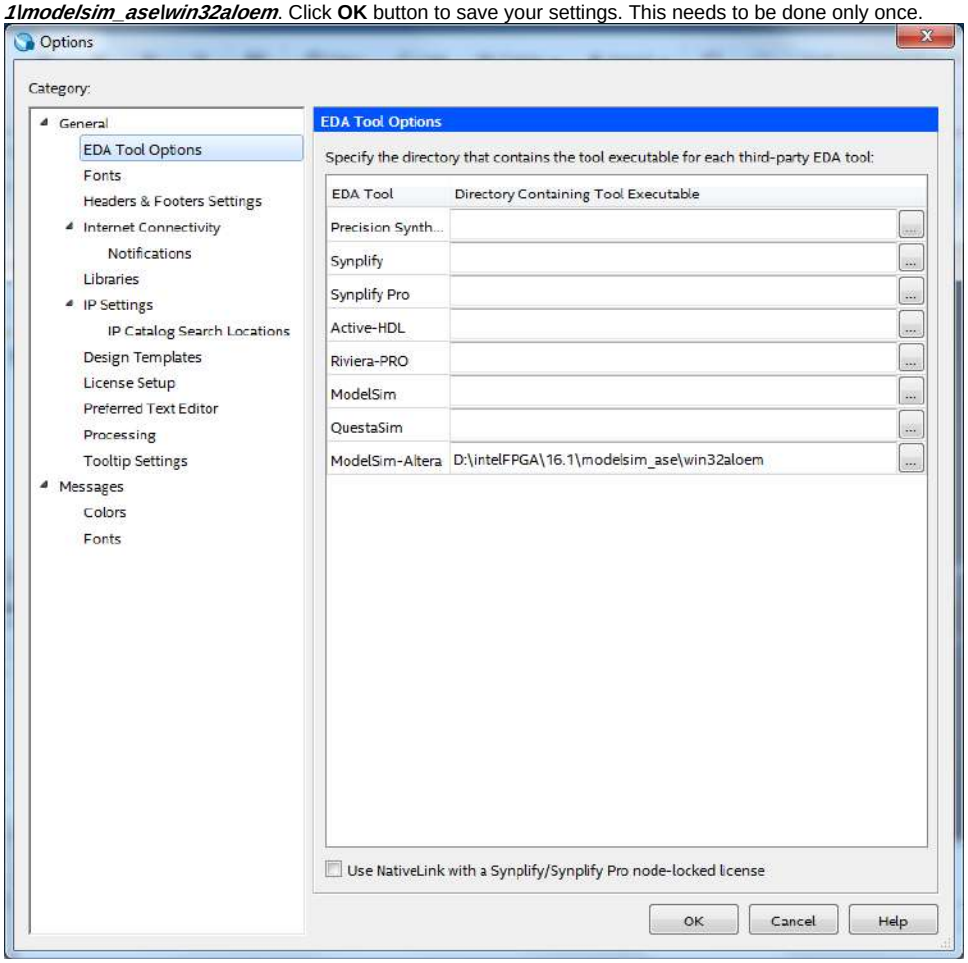
The instructions below were written for software version 16.1. Latest version is 19.4. Feel free to download either version, but note that there may be some discrepancies with the instructions provided below if you decide to use version 19.4.

Getting Started

1. Launch Quartus Prime 16.1. You should see an application window that looks like this:

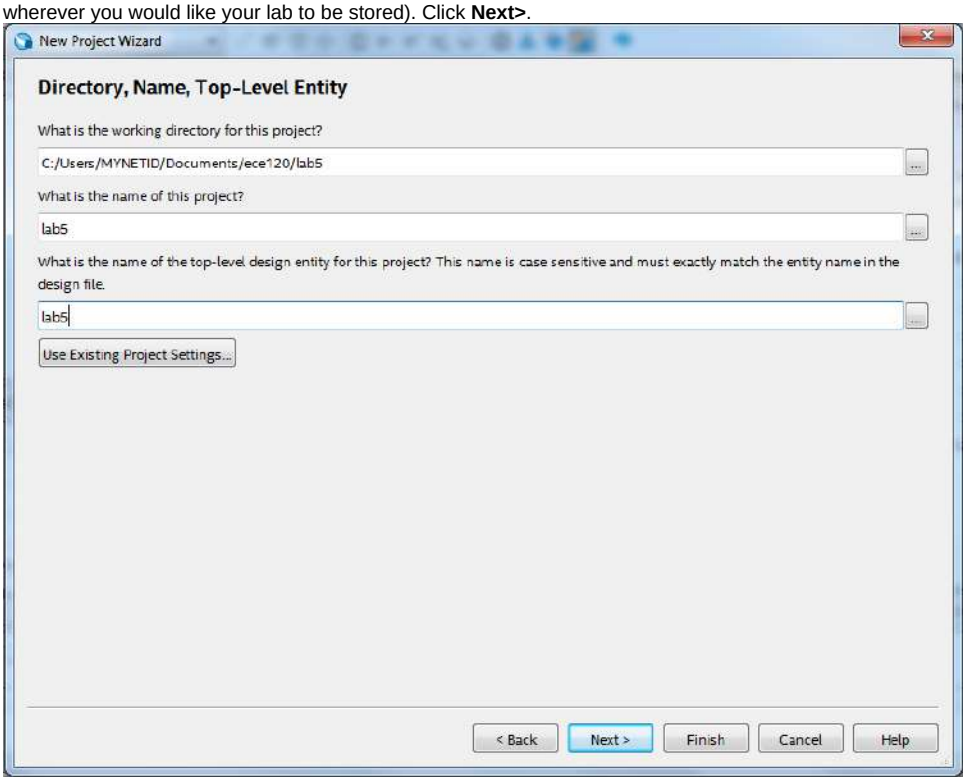


2. In **Tools -> Options** menu, setup path to the ModelSim tool as shown below. Specifically, go to **General / EDA Tool Options** section and specify the location of ModelSim-Altera to be whatever location you have installed it in. For example, it could be ***D:\intelFPGA\16.1\modelsim_ase\win32aloem***. Click **OK** button to save your settings. This needs to be done only once.

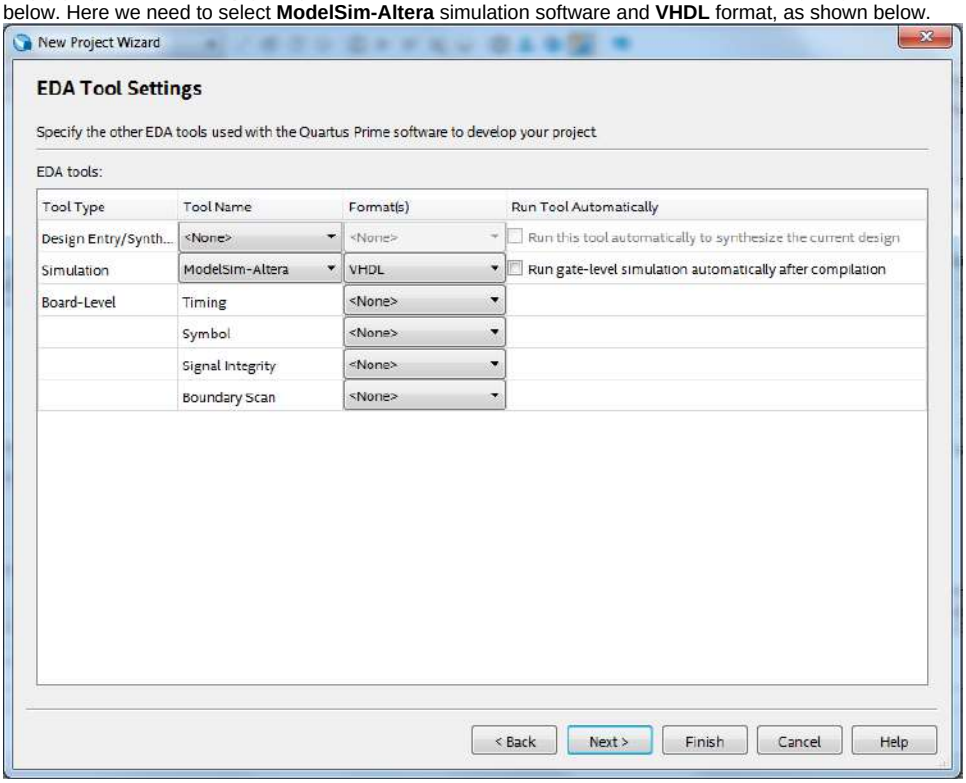


Creating a new project

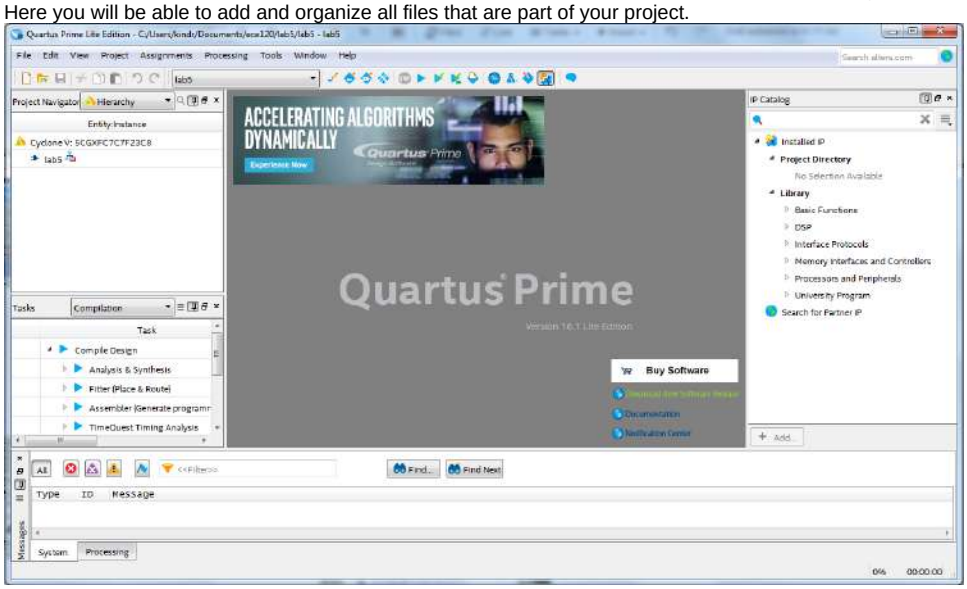
1. Inside Quartus Prime, select **File -> New Project Wizard** on the menu and click **Next** button in the first *Introduction* dialog box. You should see the following *Top-Level Entry* dialog box. Name your project "lab5" and change the Directory to **C:/Users/MYNETID/Documents/ece120/lab5** (or wherever you would like your lab to be stored). Click **Next>**.



2. The next screen will ask you to confirm the name of your project and the directory to store your project. Double check everything and click **Yes** if everything is correct. This will bring you to the next dialog box where you will be asked to choose between an existing project template or an empty project. Select **Empty Project** and click **Next>**. Click **Next>** a few more times until you get to the **EDA Tool Settings** window shown below. Here we need to select **ModelSim-Altera** simulation software and **VHDL** format, as shown below.



3. Once the simulation tool and format are selected, click **Finish** button. You should see the following main window for the *Quatrus Prime* software. Here you will be able to add and organize all files that are part of your project.



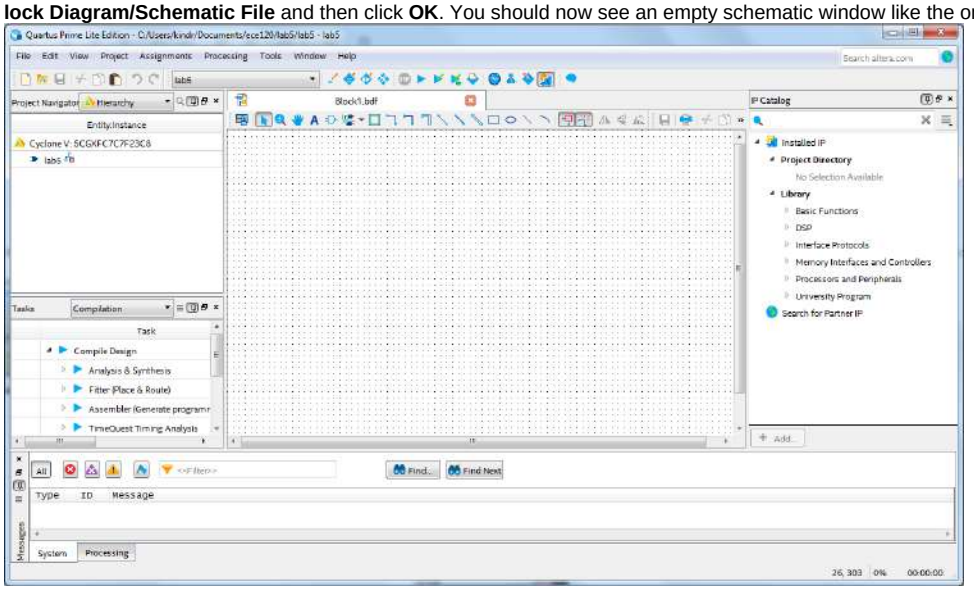
Creating new circuit

In this section, we will show you how to create a new circuit. In particular, we will show you how to implement the XOR function by using AND gates, OR gates, and NOT gates. The XOR function has two inputs *A* and *B* and one output *f*. The XOR function can be thought of as an odd/even detector: *f* is 1 when there are an odd number of 1s in its inputs and 0 when there are an even number of 1s in its inputs. For example, the first row has zero 1s in the inputs, so the output is 0, but the second row has one 1 in the inputs, so the output is 1. The relationship between the inputs and outputs is formally defined by the truth table below where the inputs are listed in the two left columns and the output is in the column on the right.

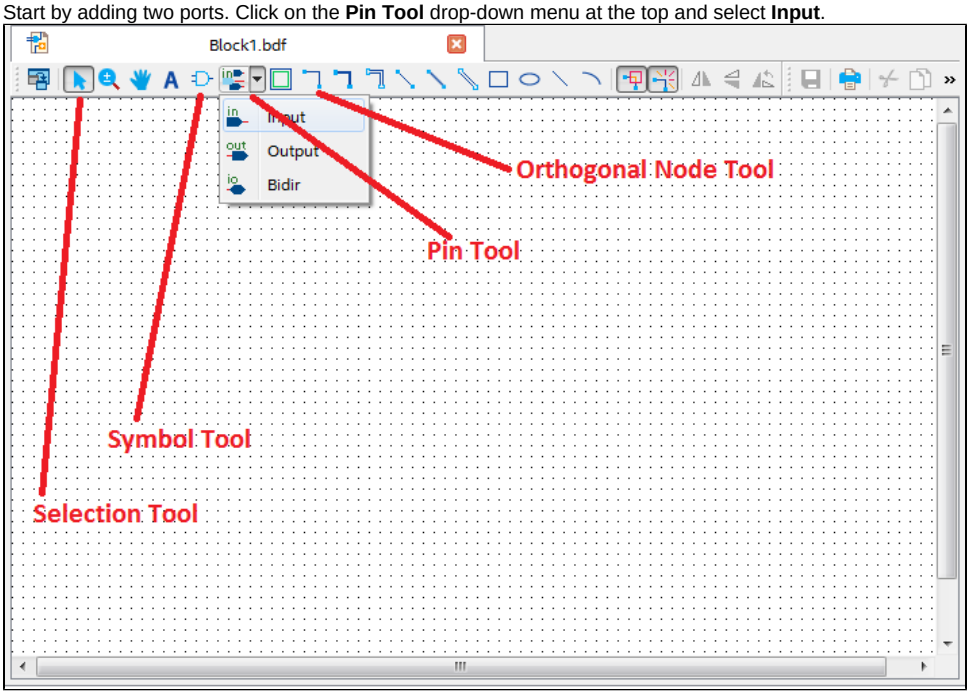
A	B	***f = A XOR B***
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Create a new block diagram and draw your circuit

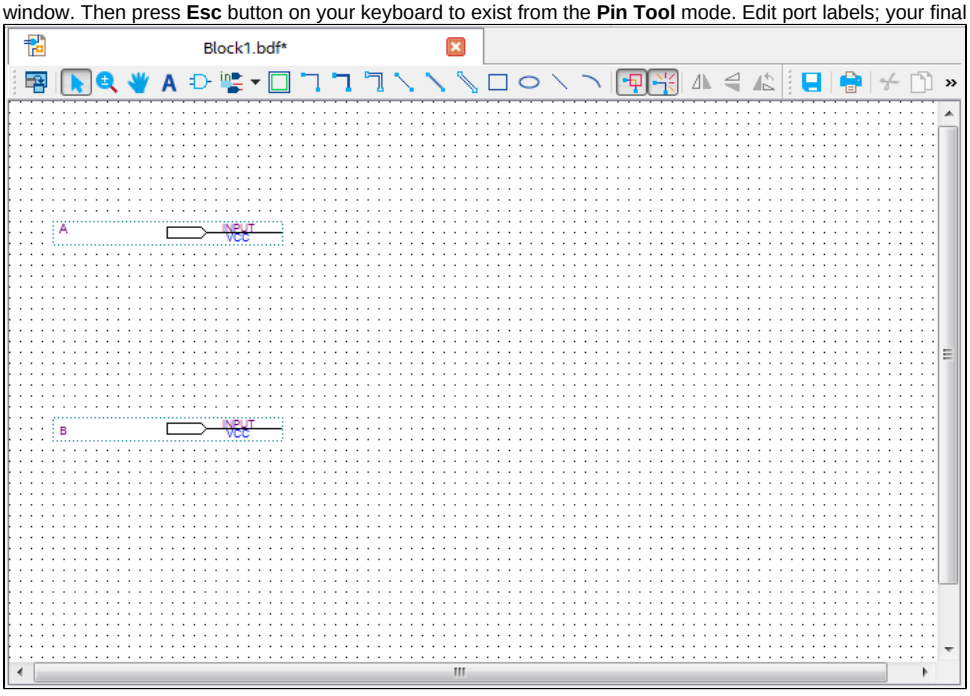
1. In the main window, click the **New** button on the top left, or select **File -> New** menu option. New window will appear with a list of options. Select **B lock Diagram/Schematic File** and then click **OK**. You should now see an empty schematic window like the one below.



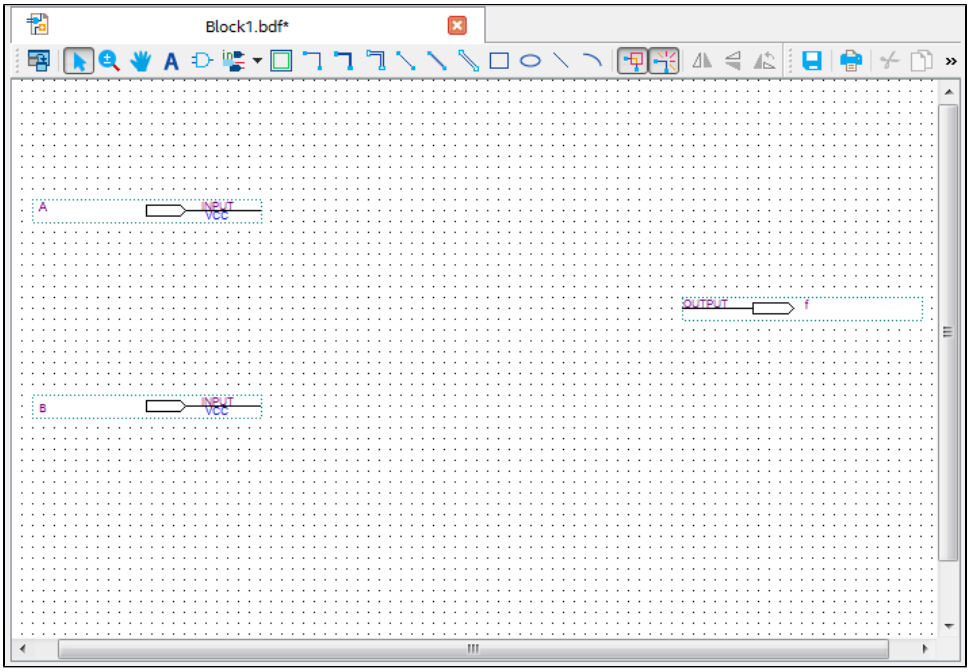
2. Start by adding two ports. Click on the **Pin Tool** drop-down menu at the top and select **Input**.



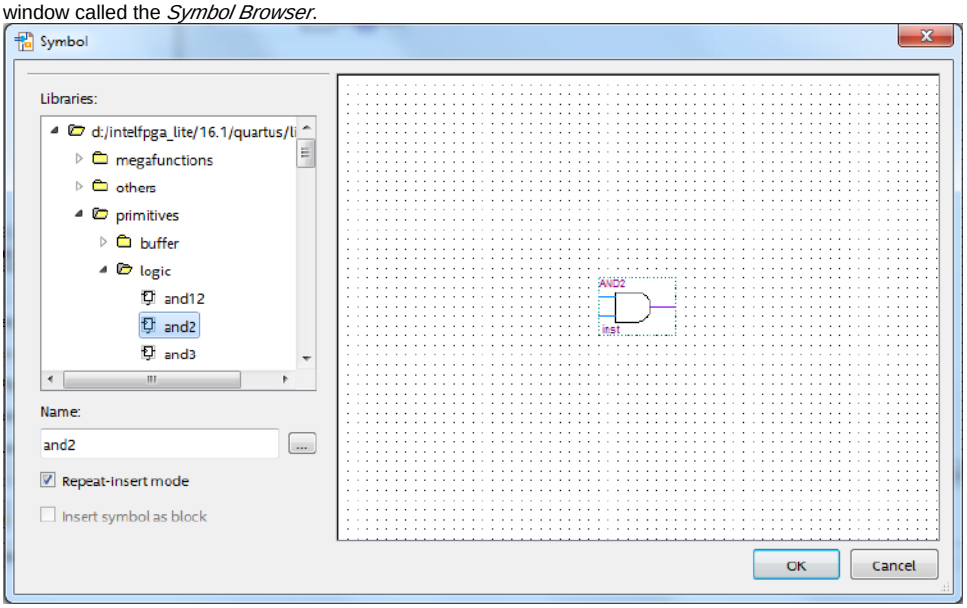
3. Your mouse cursor will change to *input port* symbol. Add two input ports, by clicking on two different places on the left side in the schematic window. Then press **Esc** button on your keyboard to exist from the **Pin Tool** mode. Edit port labels; your final result should look as follows:



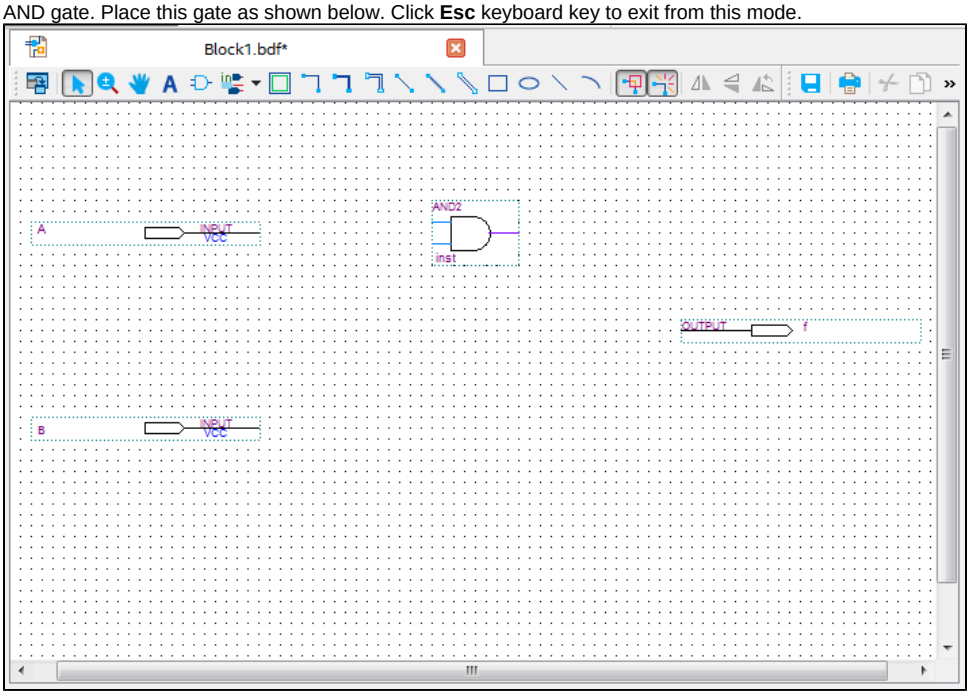
4. Add an output port. Click on the **Pin Tool** drop-down menu at the top and select **Output**. Place a single output port on the right side of the schematic window and edit its label.



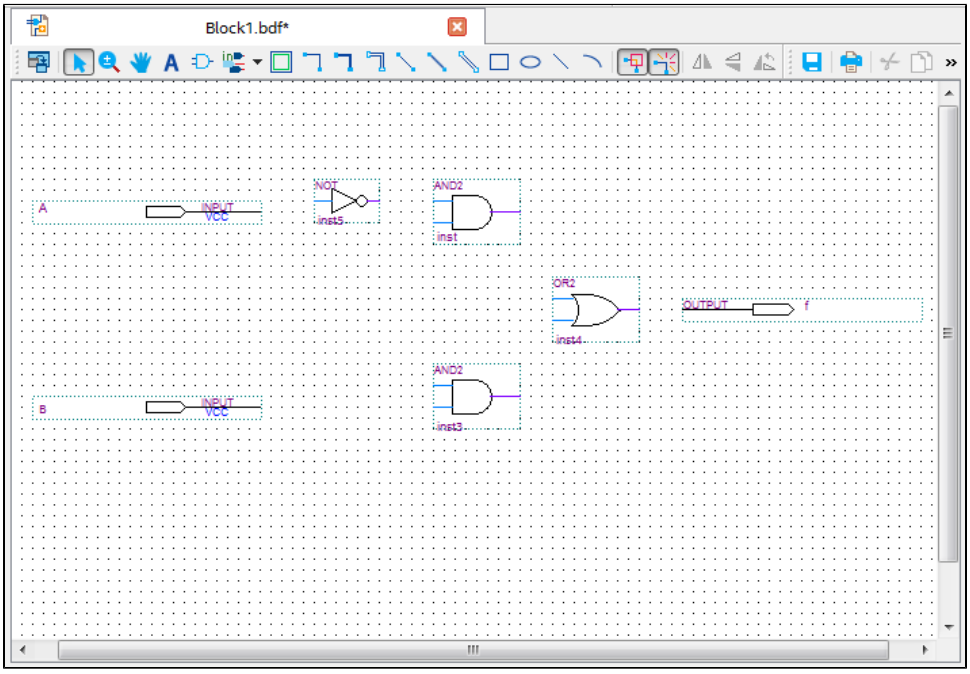
5. Add an AND gate.
   a. Click on the **Symbol** tool icon left to the **Pin Tool** drop-down menu. The icon looks like a 2-input AND gate. You should see a new window called the *Symbol Browser*.



b. In the **Libraries** section of the *Symbol Browser* navigate to find the **2 input AND gate** (under **primitives/logic** section) as shown above. Click on **and2** and then click **OK** button. You will be returned to the schematic window and your mouse cursor will look like an AND gate. Place this gate as shown below. Click **Esc** keyboard key to exit from this mode.



6. Now add one more AND gate, an inverter (also called a NOT gate), and an 2-input OR gate (which we will call an OR gate from now on) as shown below.

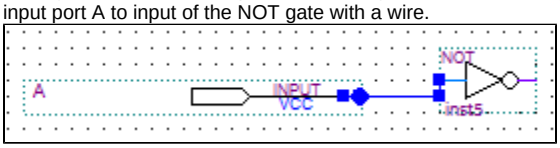


7. Add a wire.

a. Select **Orthogonal Node Tool**; your mouse pointer will change to look like the icon of the Orthogonal Node Tool.

b. Place mouse pointer over the right end of INPUT A port and **press and hold** left mouse button. You will see a small square appearing at the end of the INPUT A port.

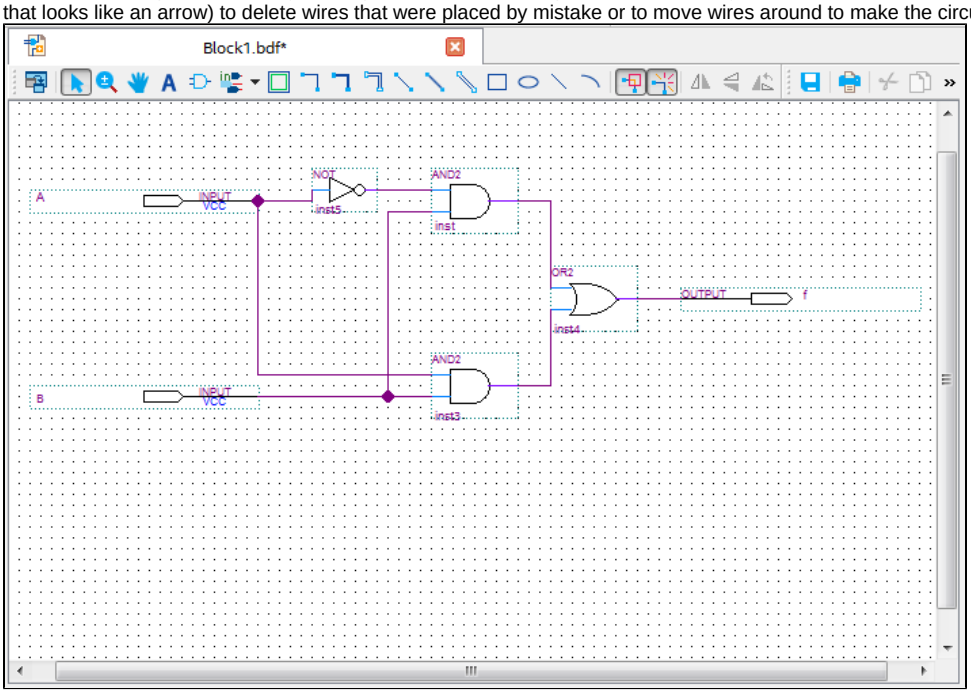
c. Move your mouse to the input of the NOT gate until a similar square appears. Release the left mouse button. You now have connected input port A to input of the NOT gate with a wire.



d. You should notice that wires can be drawn only with horizontal or vertical components and the schematic designer will make its best guess for where to put some 90 degree turns rather than make diagonal wires.

e. As long as you don't press **Esc**, each mouse press will either start a wire (if you are not currently drawing a wire) or end a wire (if you click on the port of another component). If you click on an existing signal, it will create a new wire that carries the same signal to a different destination. A dark circle at an intersection shows that the two wires are connected and carry the same signal. Intersections without this black circle are not connected.

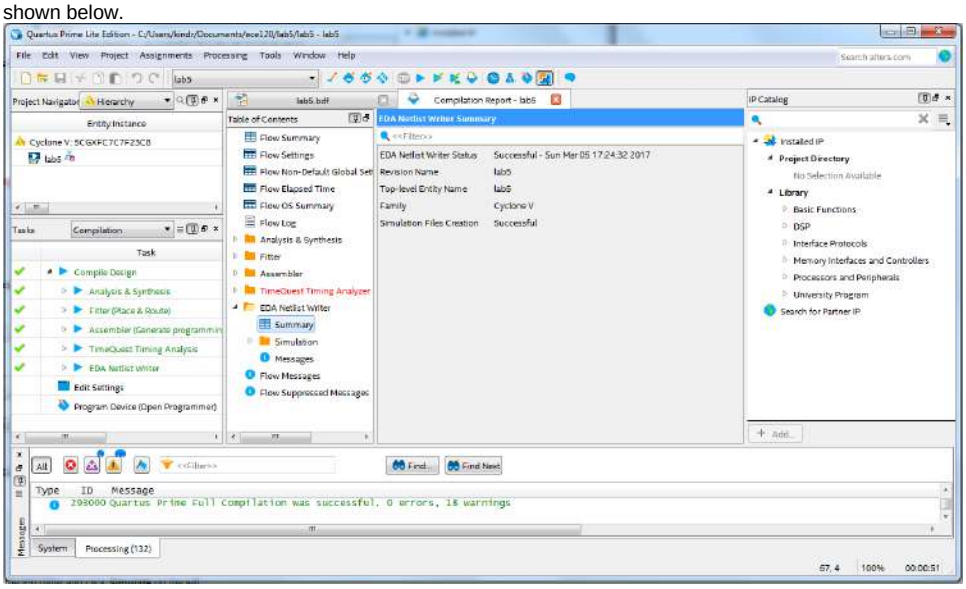
8. Add the remaining wires so that your diagram resembles the figure below. You may need to use **Selection** tool (second from the left end button that looks like an arrow) to delete wires that were placed by mistake or to move wires around to make the circuit look "cleaner".



9. Save your design. Notice the ***** by the name of your file at the top of the window. That ***** means that your design has not been saved since its last modification. Save your schematic either with the pull down menus or by pressing **Ctrl** and **s** at the same time. Save your schematic early and often to avoid losing your work in case Quartus software crashes.

Generate HDL for your circuit

1. In order for the simulator to simulate your circuit, the schematic you drew needs to be converted into a language that the simulator understands. The simulator uses a language called a VHSIC Hardware Description Language (VHDL). Click on **Process->Start Compilation** menu item. It will take a while, but eventually the process will stop and you will be able to navigate to **EDA Netlist Writer Compilation Report** as shown below.



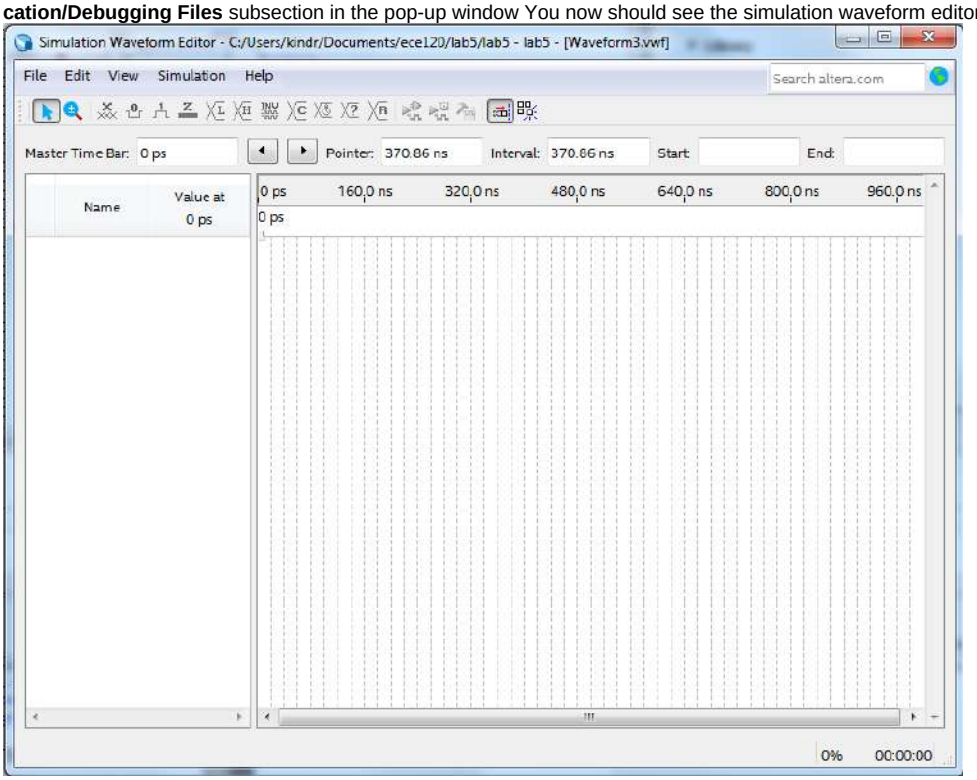
2. If you encounter errors during the compilation, review and fix them. Ignore TimeQuest Timing Analyzer errors. But there should be no errors generated by the EDA Netlist Writer process. You may need to refer to Quartus Intro tutorial for more details or ask questions in office hour.

Simulating your circuit

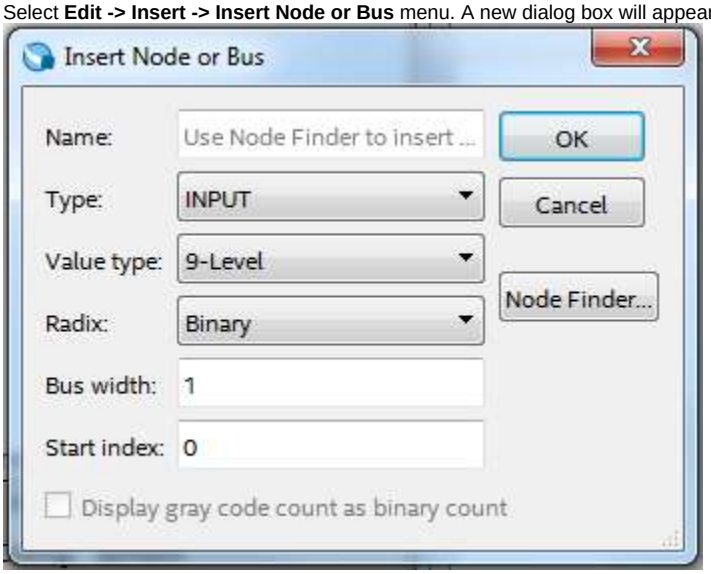
In this section, we will learn how to simulate a circuit so that we can make sure that it behaves as we expect.

Setup the Simulation

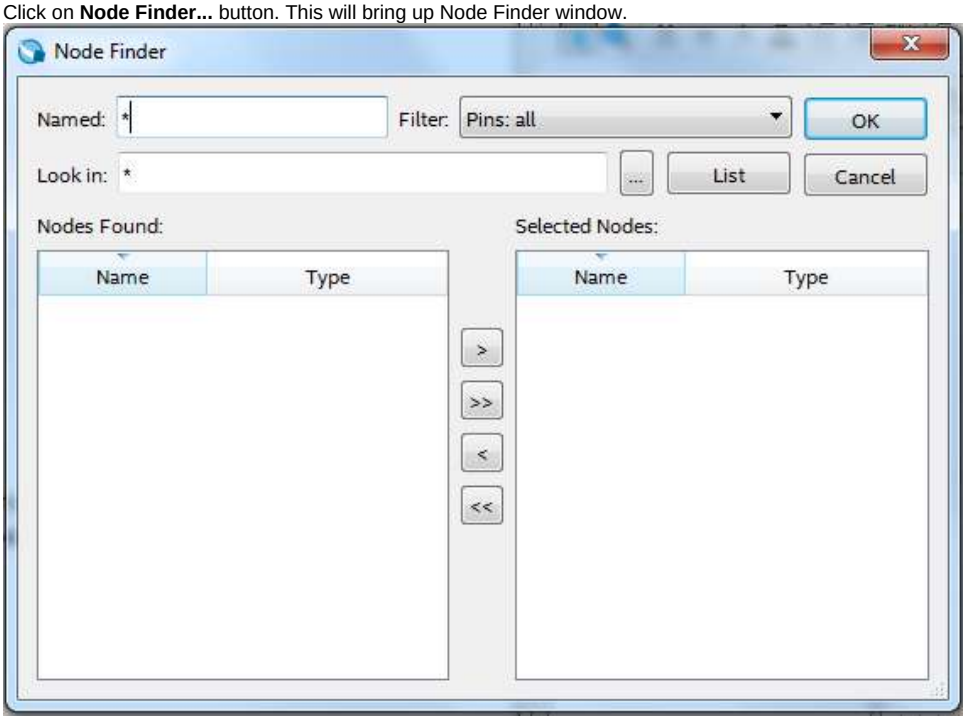
1. Start the simulator waveform editor. Select **File -> New** menu in the main window and then select **University Program VWF** file type under **Verification/Debugging Files** subsection in the pop-up window You now should see the simulation waveform editor.



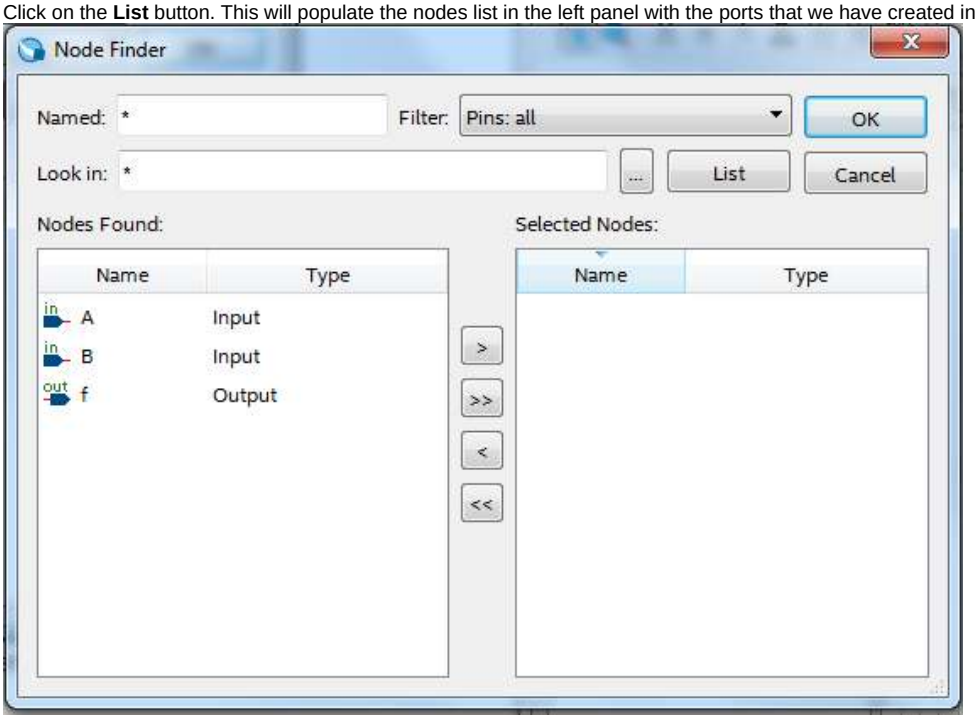
2. Choose which signals to simulate.
    a. Select **Edit -> Insert -> Insert Node or Bus** menu. A new dialog box will appear.



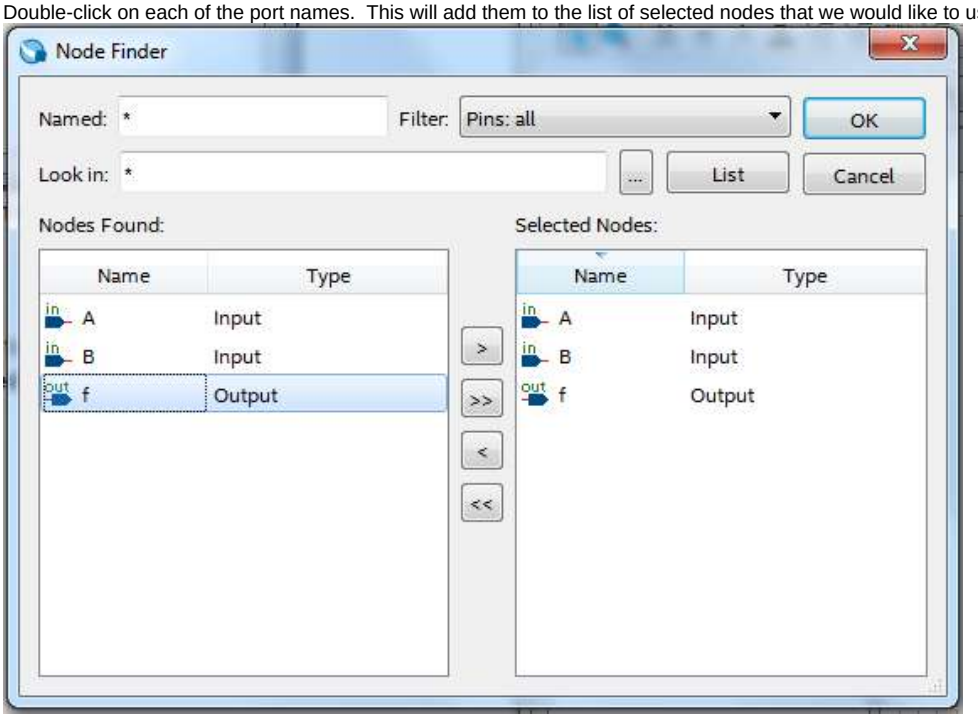
b. Click on **Node Finder...** button. This will bring up Node Finder window.



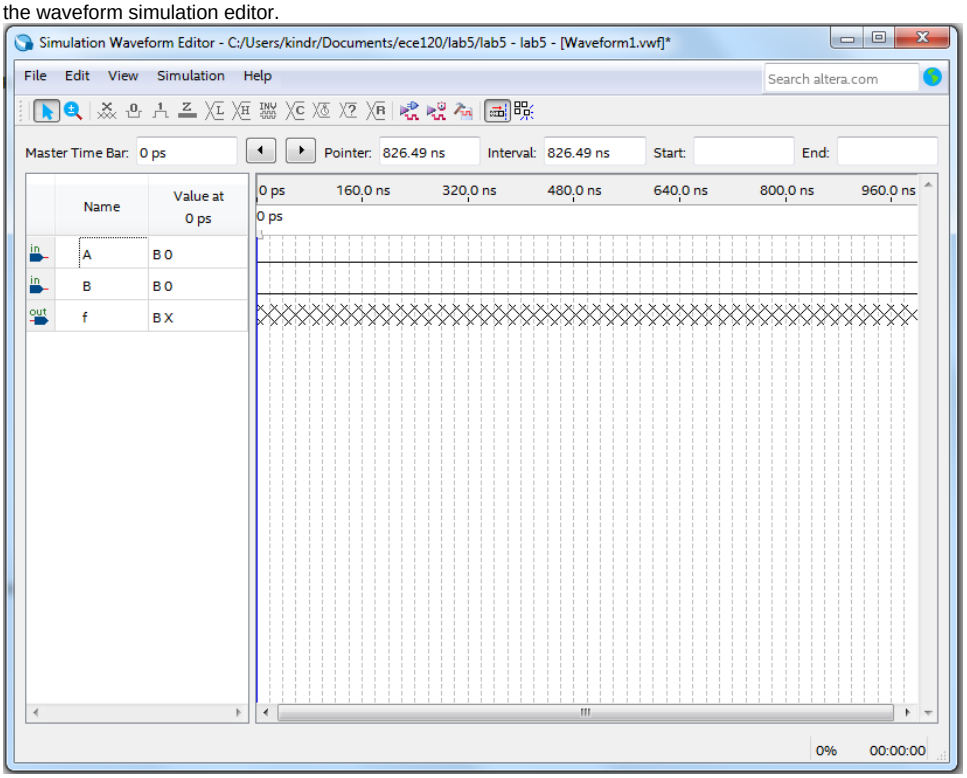
c. Click on the **List** button. This will populate the nodes list in the left panel with the ports that we have created in our design.



d. Double-click on each of the port names. This will add them to the list of selected nodes that we would like to use in our simulation.



e. Click OK button in this window and then OK button in the **Insert Node or Bus** window. You will see that the three signals were added in the waveform simulation editor.

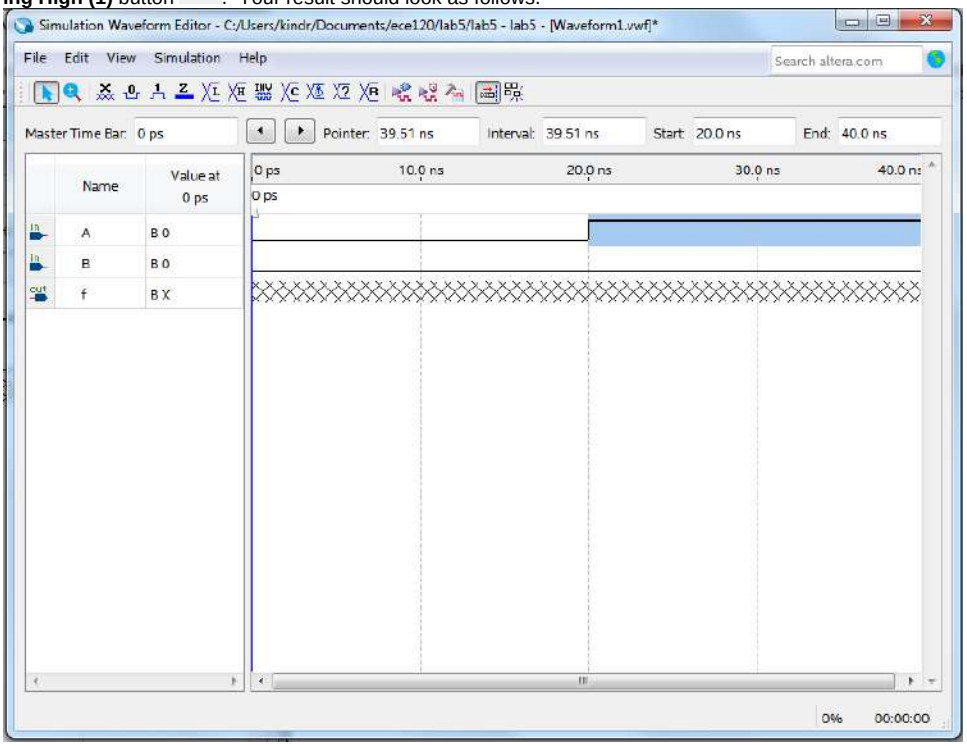


3. Setup simulation parameters. This includes setting up simulation duration and editing input signals waveforms.

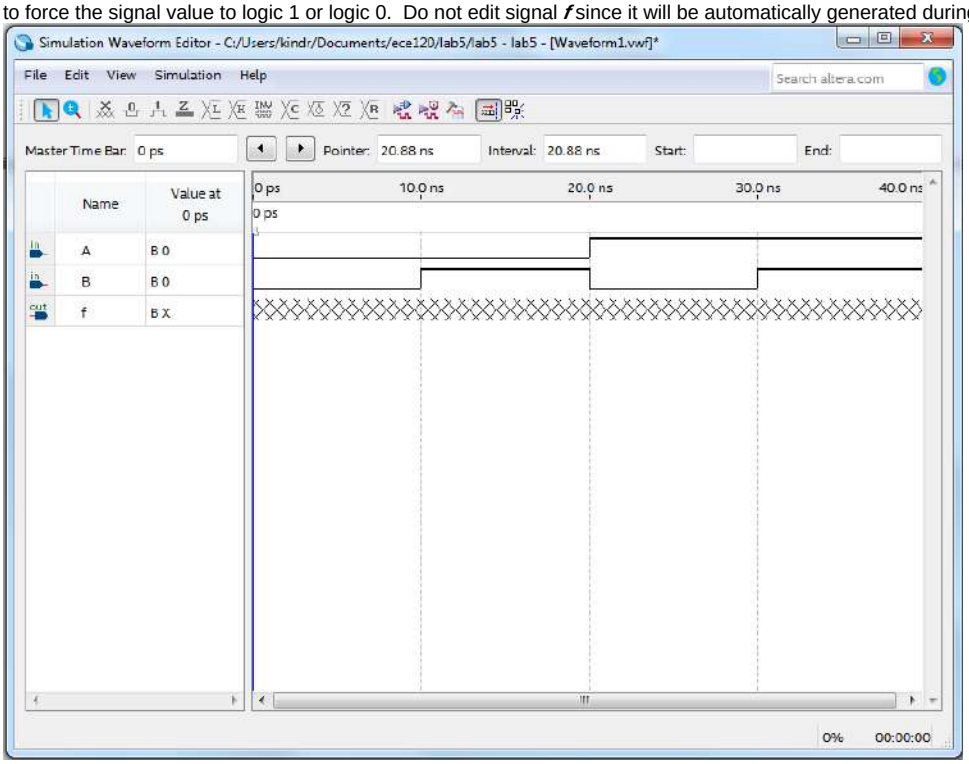
a. Click on **Edit -> Set End Time** menu in the Simulation Waveform Editor. This will bring up a small dialog box in which you need to set the simulation end time to 40 ns. You will notice that the time line at the top of the waveform window changed to 0-40 ns scale.

b. Specify the logic values to be used for the input signals A and B during simulation. First, position your mouse on top of signal A waveform starting at 20 ns. Next, push and hold left mouse button and move the mouse to he right until a blue rectangle covers the entire 20-40 ns interval. Release the mouse button. You have selected part of the signal A waveform which you can now edit. Press **Ctrl**

+ **Alt** + **1** keyboard keys simultaneously. This will force selected signal value to be high. You can also achieve this by clicking on the **Forcing High (1)** button  . Your result should look as follows.



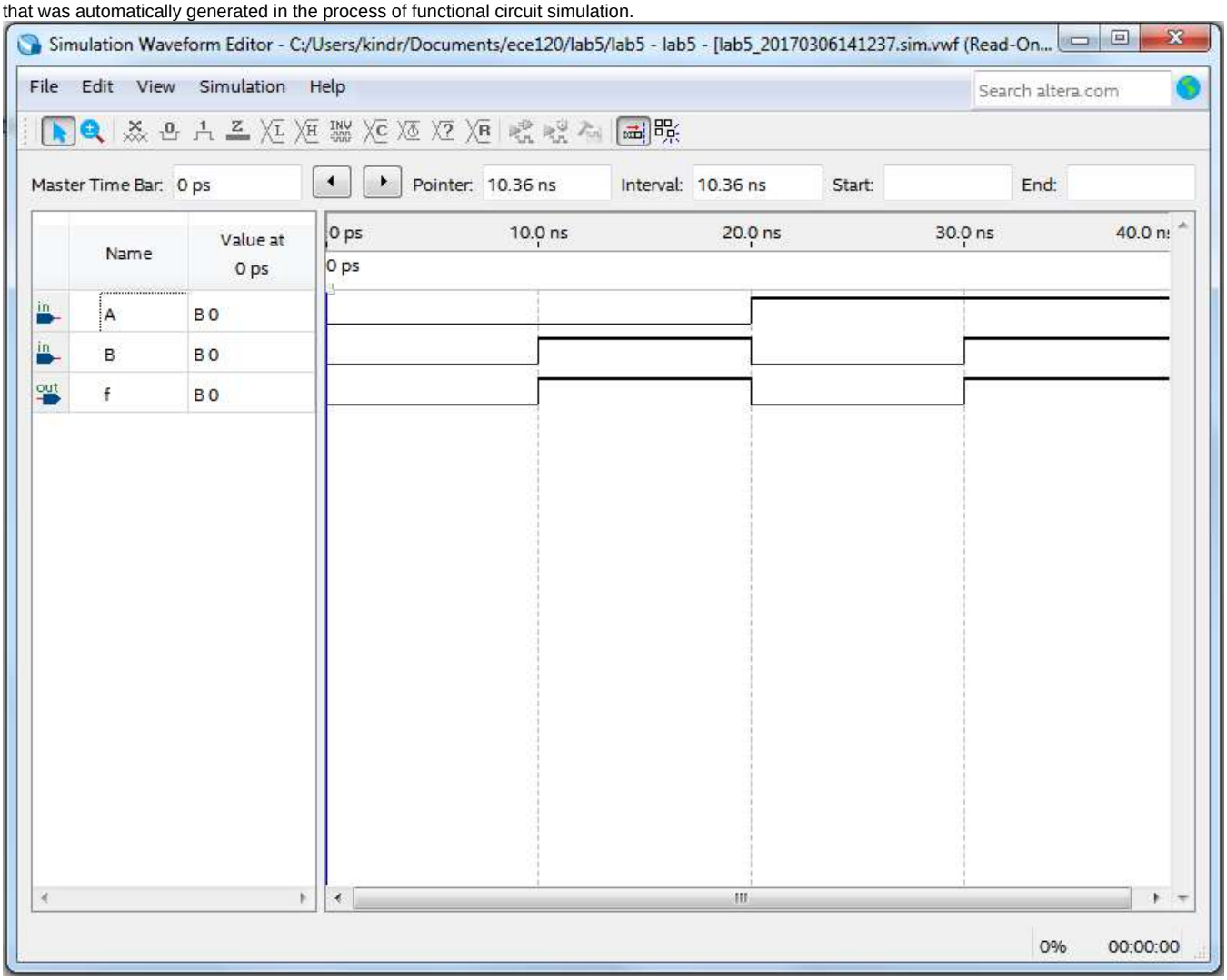
c. Edit waveform for signal B as shown below. Buttons  and  or keyboard shortcuts **Ctrl** + **Alt** + **1** and **Ctrl** + **Alt** + **0** can be used to force the signal value to logic 1 or logic 0. Do not edit signal ***f*** since it will be automatically generated during the simulation.



4. Save your work: select **File -> Save** menu in the waveform editor window and click on **Save** button in the popup window. You can change the file name, if you want.

Run the Simulation

Select **Simulation -> Run Functional Simulation** menu in the waveform editor window. This will bring up Simulation Flow Progress window which will quickly disappear. Next, you will see another simulation waveform editor window that will contain both the signals **A** and **B** you have set and the signal ***f*** that was automatically generated in the process of functional circuit simulation.



This is it, you have successfully simulated the circuit that you have designed in the schematics editing tool.

Evaluate whether your circuit behaves correctly

Compare your waveform to the truth table for the XOR function.

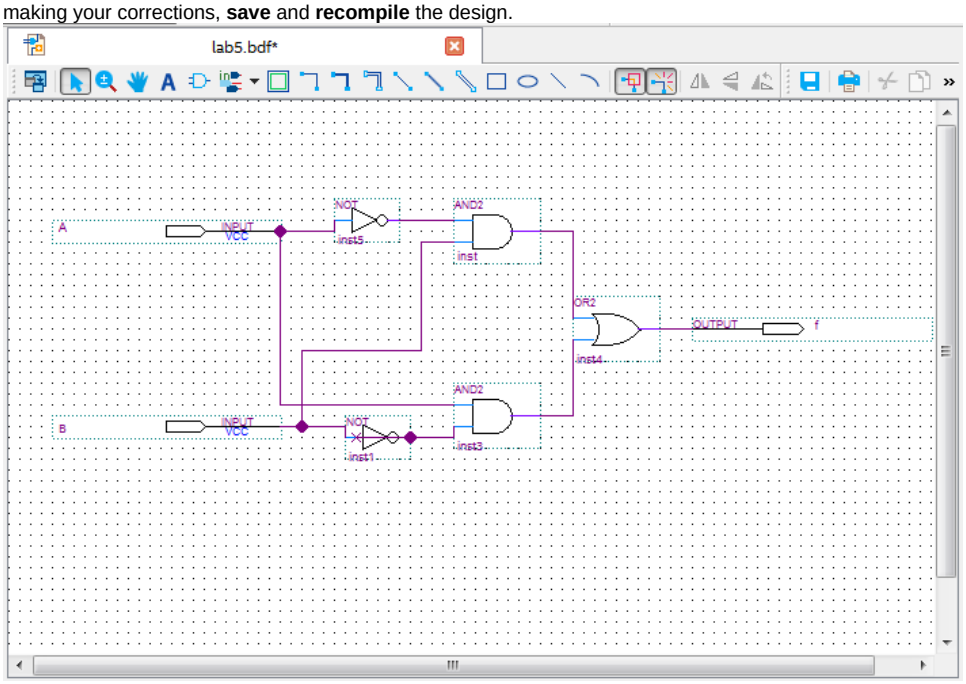
A	B	***f* = *A* XOR *B***
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

- In the waveform, we can see that for 0-10 ns, the values of *A*, *B*, and *f* are all 0 (A=0, B=0, f=0). This behavior matches the first row of our truth table.
- We can see that for 10-20 ns, A = 0, B = 1, f = 1. This behavior matches the second row of our truth table.
- We can see that for 20-30 ns, A = 1, B = 0, f = 0. This behavior does not match the third row of our truth table!
- We can see that for 30-40 ns, A = 1, B = 1, f = 1. This behavior does not match the fourth row of our truth table!

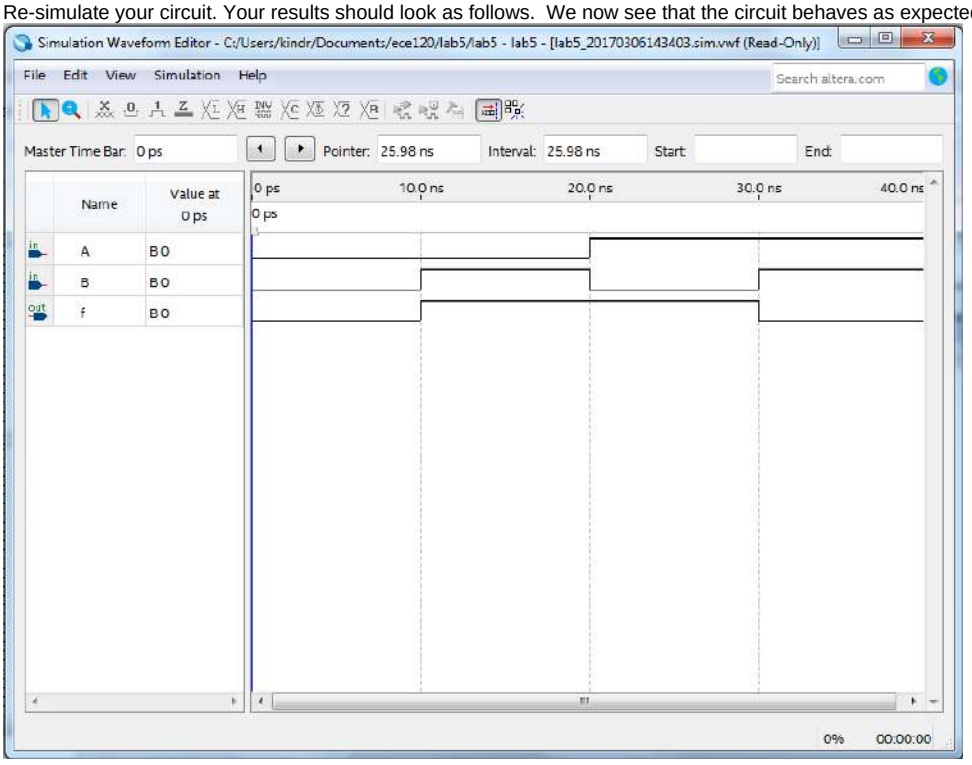
Debug and correct your circuit

From this comparison, we can see that our circuit does not behave correctly. We can debug our circuit by determining the Boolean expressions for our desired XOR function and what the waveform shows that we implemented. The XOR function: $f = A'B + AB'$. Our waveform function: $f = A'B + AB$. This analysis shows us that one of the AND gates in our circuit has the wrong inputs. We need to change the A *AND B* gate to be an *A AND B'* gate.

1. Correct your circuit. Close both waveform editing windows and open your circuit schematic. Add an inverter after the **B** input port so that we can create our *A AND B'* gate and reconnect the wires as shown below. Make sure that your input ports are all named correctly. When you are done making your corrections, **save** and **recompile** the design.



2. Re-simulate your circuit. Your results should look as follows. We now see that the circuit behaves as expected.



Lab 5 assignment

Now that you know how to use the tools, let's build a full adder circuit. Full adder is one component we can use to add unsigned binary numbers together. In binary addition, we need to create both a sum output bit and a carry output bit. To better understand the roles of these bits, think about regular decimal addition. If we add 5+7, we would get a sum digit of 2 in the ones place and we would carry a 1 into the tens place to get the number 12. Similarly, if we add 1+1 in binary, we would get a 0 in the ones place and we would carry a 1 to the twos place to get the number 10 (2 in decimal).

The full adder adds three input bits together: *a*, *b*, and carry-in (*Cin*). It also has two output bits: the sum (*s*) and the carry out (*Cout*). If we arrange *Cout* and *s* as two bits of an unsigned binary number, then this binary number tells us how many 1s are in the inputs. For example in the fourth row of the truth table, <*a,b,Cin*> = <0,1,1>, so <*Cout,s*> = <1,0> or 2 in decimal. Similarly in the fifth row of the truth table, <*a,b,Cin*> = <1,0,0>, so <*Cout,s*> = <0,1> or 1 in decimal.

a	*b*	*Cin*	*Cout*	*s*
0	0	0	**0**	**0**
0	0	1	**0**	**1**
0	1	0	**0**	**1**
0	1	1	**1**	**0**
1	0	0	**0**	**1**
1	0	1	**1**	**0**
1	1	0	**1**	**0**
1	1	1	**1**	**1**

What to do

1. Create a new project and create a new schematics design that implements the above truth table for functions **Cout** and **s**.
2. Create a simulation waveform that goes over all 8 input combinations, simulate the circuit, and verify that **Cout** and **s** functions behave as expected.

What to turn in

1. Printout of the schematic diagram labeled with your name and numerical ID.
2. Printout of the simulated waveform.

Merge these two pages together and upload a single pdf file to Blackboard. **This will constitute your Lab 5 submission.**