**ECE 2504 – Introduction to Computer Engineering**

**Design Project 0: Using your Hardware Design Tools**

1. **Preliminaries**

*Honor Code Requirements*

You must complete this project individually. Do not discuss any aspect of your solution or approach with any other student. Copying or using *any* other student’s design or responses is a violation of the Virginia Tech Honor Code, and will be prosecuted as such. You may discuss general features of Quartus, your DE0 Nano Board, and your parts kit with other students.

Direct *all other questions* to your GTA or to your instructor.

*Objective*

After completing this project, you should be able to use the hardware design tools for this class. You will learn how to use Quartus to represent circuits and implement them on your DE0 Nano Board. You will also learn how to use Qsim to simulate basic circuits.

*Preparation*

You will carry out a series of activities in Quartus that will culminate in an implementation exercise. You will also design a circuit that you will implement on your DE0 Nano Board. A GTA will validate your design, but you will not have to take your board to the CEL for validation. Instead, you will have to upload your design files and other project materials to your course Canvas site, and complete a set of questions on your course site.

Before starting the lab assignment, you should:

* Read this specification.
* Follow the instructions in the document “Installing Altera CAD Tools” to install Quartus. Review the document if you have installation problems.
* Read the document “Creating and Simulating Designs in Quartus” for information on creating circuits and simulating them. This specification also reviews much of the same information.

You will find both of these documents in the ECE 2504/ECE 3544 Lab Manual.

*Materials*

You will need access to a computer that can run the **Quartus Prime Lite Edition software package**. You will also need the **DE0 Nano Board** and the **ECE 2504 Parts Kit**.

If you have questions on the use of your trainer kit or your parts, see your instructor or a CEL GTA.

The project materials include two Quartus project archive files. **P0Simulation.qar** contains implemented versions of the circuits described in the procedures and assignment questions. **P0Implementation.qar** contains the framework for the circuit that you will implement on your DE0 Nano board. You will invoke Qsim inside of Quartus to simulate the circuits.

# Getting Started with Quartus and Qsim

The two schematics that you will need to answer the project questions are contained in the file **P0Simulation.qar**. You will not need to create any circuit schematics for this part of the project. Your task is to open this archive and perform simulation. This section will explain how to perform simulations in Qsim; the next section will discuss the purpose of each schematic and what you need to record for this project.

1. Create a working folder for Project 0, e.g. c:\ece2504\Project0. *Place your working folder in a path that does not contain any spaces.*
2. Copy **P0Simulation.qar** to your working folder.
3. Open Quartus. If the “Getting Started with Quartus” window pops up, you may close it.
4. In Quartus, choose **File** > **Open Project**, and select **P0Simulation.qar** from your working folder. Click **Open**.
5. If prompted, click **OK** to restore the archived project. Once you have restored an archive, you can use the appropriate project file (\*.qpf) to open the project in future instances. *If you double-click the archive after restoring it for the first time, you could overwrite your saved work.*
6. In the upper left, you will find the Project Navigator window. It should default to showing the Hierarchy tab. Double-click **P0Simulation**. A schematic named **P0Simulation.bdf** will pop out.

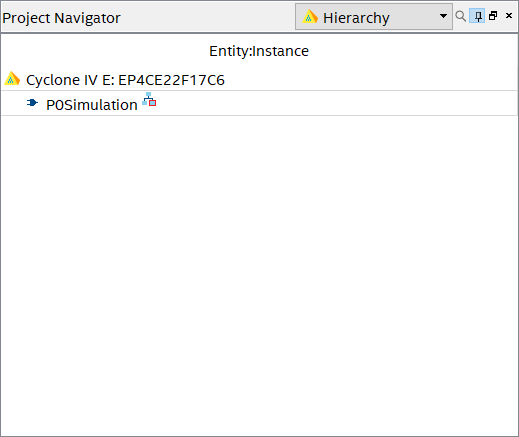
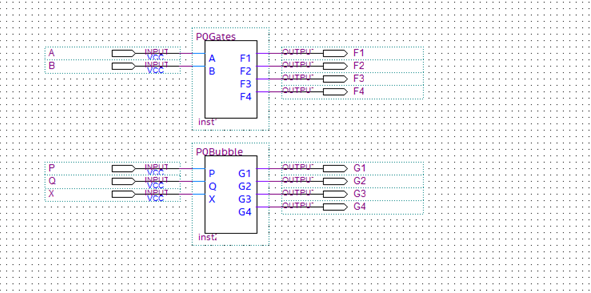


Figure 1.1 The *Hierarchy* Tab of the *Project Navigator* Window

1. There are two components in **P0Simulation.bdf**. Each of the two components corresponds to a section of the project questions.



These blocks contain (ject.)orm that you generate for your project might end up looking different. Follow the input/output correspondence for your pject.)orm that you generate for your project might end up looking different. Follow the input/output correspondence for your pimplementation details of the next lower level.

Outputs

Inputs

Figure 1.2: The *P0Simulation* window. Yours may look slightly different.

***Tip****: If you are using a high resolution screen, you may have trouble seeing the port labels shown in Figure 1.2. You can overcome this issue by either right-clicking the label and viewing the Properties, OR reducing your screen resolution, restarting/signing out of Windows, and restarting Quartus.*

Right-click **P0Gates** and choose **Open Design File**. A second tab called **P0Gates.bdf** will open showing the internal connections between sets of NAND gates. (Another way to do this is to choose **Files** in the Project Navigator, then double-click **P0Gates.bdf**.) You will see four schematics in the **P0Gates.bdf** tab having inputs A and B and outputs F1, F2, F3, and F4. These correspond to the inputs and outputs of the truth tables in Part I. We will use them in the simulation.

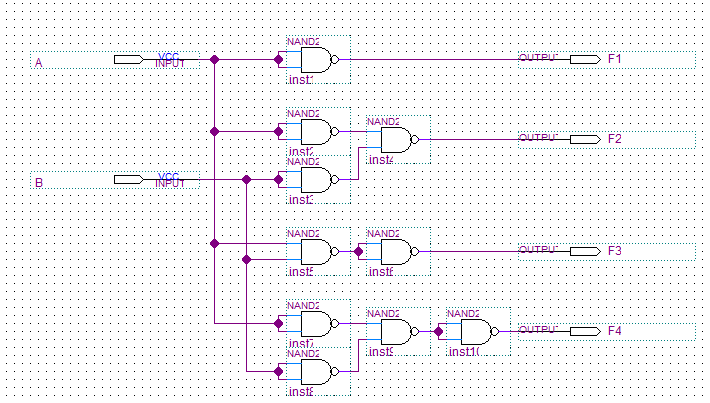


Figure 1.3: The *P0Gates* Window. Yours may look different

1. The schematics in P0Simulation are already complete; you do not need to modify them. However, you should ensure that the simulator is using the most up-to-date version of your circuit schematic by compiling the current design. Go to the **Tasks** window in Quartus and double-click **Analysis & Synthesis** under **Compile Design**. (Or choose **Processing** > **Start Compilation**.) Wait for compilation to complete before you continue.

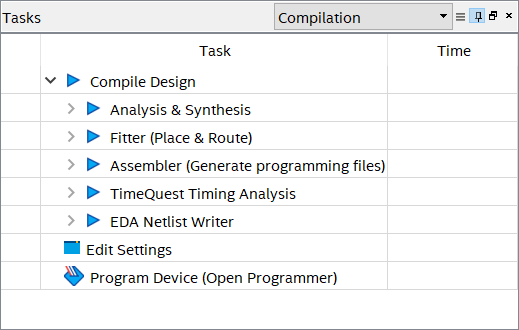
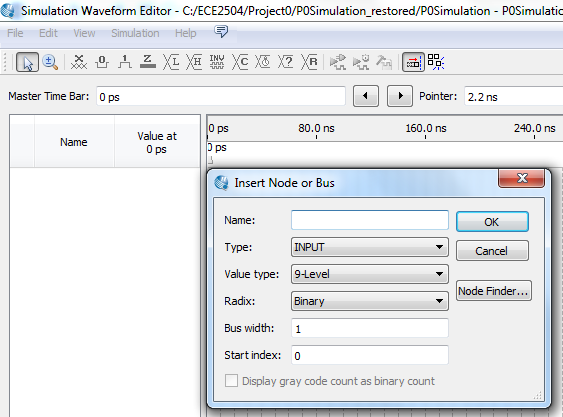


Figure 1.4: The Tasks Window.

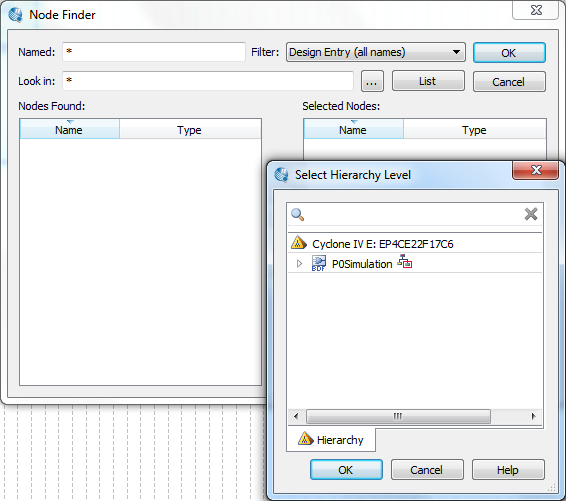
1. To begin simulation, choose **File > New**, then choose **University Program VWF** under **Verification/Debugging Files**. This will invoke the Simulation Waveform Editor.
2. Double-click the blank area as below to open the **Insert Node or Bus** dialog box. Click **Node Finder…**



Double-click in here

Figure 1.5: The Simulation Waveform Editor

1. Select **Design Entry (all names)** from the Filter drop-down menu. Click **…** to select the hierarchy. Select **P0Simulation** in the Select Hierarchy pop-up window and click **OK**.



Select **Design Entry (all names)**

Click **…** to select hierarchy

Select **P0Simulation**

Figure 1.6: The Node Finder

1. Click the **List** button to display the list of nodes. If no nodes appear, you should re-compile the design. Close the waveform file, then repeat these steps starting with Step 8.
2. Under Nodes Found, select the nodes that correspond to the inputs and outputs of the logic circuit. Click  to move them to the **Selected Nodes** list and click **OK**. Press the Shift key to choose all items between the first and last items on a list. Press the Ctrl key to select non-consecutive items on the list.

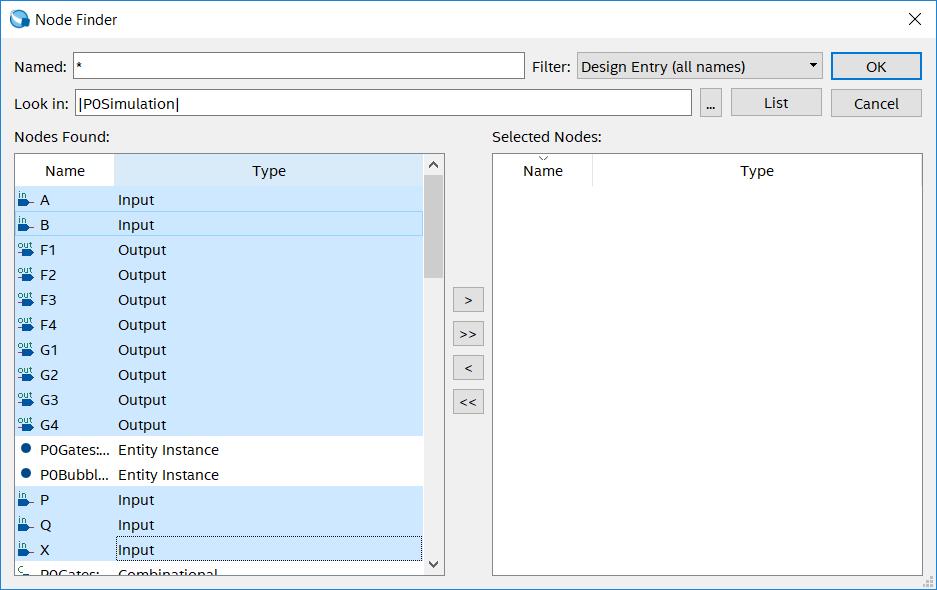


Figure 1.7: Choosing Nodes in the Node Finder

1. Click **OK** on the **Insert Node or Bus** window. You should see something like this:

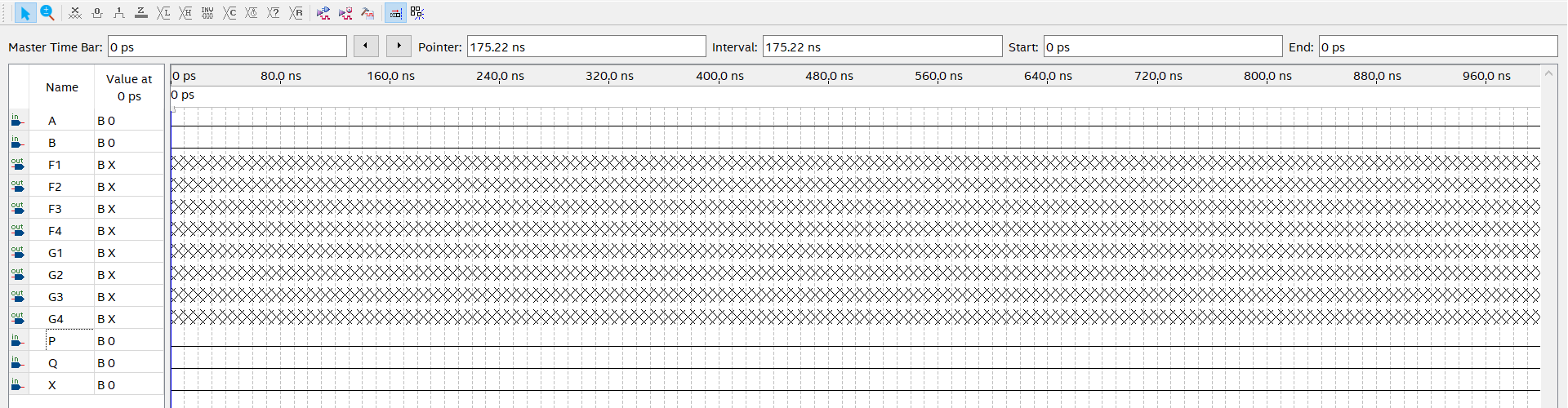


Figure 1.8: The Waveform Editor, with Nodes Inserted

Arrange signals in the window by clicking and dragging the names up or down in the window. As often as possible, you should associate the particular output signal (or set of output signals) that share a common set of inputs so that it is obvious which inputs they use. Review the original schematic and determine for yourself which output signals use A and B as inputs, which ones use P and Q and inputs, and which ones use X as the input.

1. Group inputs A and B together by selecting A and B (use Ctrl to select multiple objects), right-clicking, and choosing **Grouping** > **Group**. Name the group AB.

***Tip****: When you group signals in the future, make sure you are grouping signals that are inputs of the same circuit. You will also want to ensure that they are grouped in the order you expect (i.e. if you are calling your group AB, A should have been first, otherwise the 10 and 01 inputs will be swapped.) See the Lab Manual for more on this.*

1. Click the signal name AB to choose the entire waveform, then choose the **Count Value**  button from the toolbar. You can use the Count Value function to easily permute all of the possible combinations for a set of inputs. The default width of the timing waveform is 1 μs, or 1000 ns. Two inputs can take on four different input combinations, so each input should last for 250 ns. (We can use this principle to create simple waveforms for circuits having many inputs.)

***Tip****: In the future, DO NOT use Count Value if you don’t intend to enter all possible combinations in numeric order.*

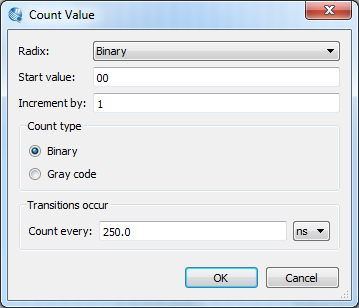


Figure 1.9: Setting Input Values using the Count Value Function

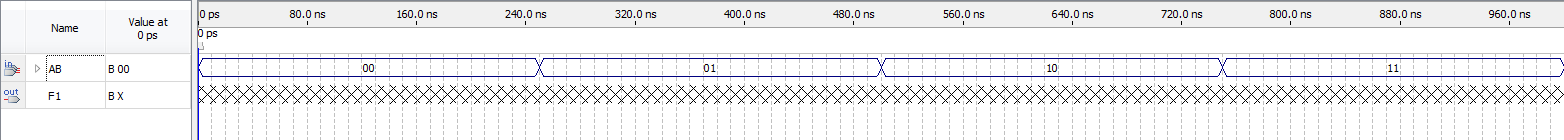


Figure 1.10: Completed Input Values

1. In the Simulation Wave Editor window, choose **File** > **Save As.** Give your file a name and click **Save**. This screenshot shows a reference input VWF file. (The waveform that you generate for your project might end up looking different. Follow the input/output correspondence provided by the circuit diagrams for your project.)

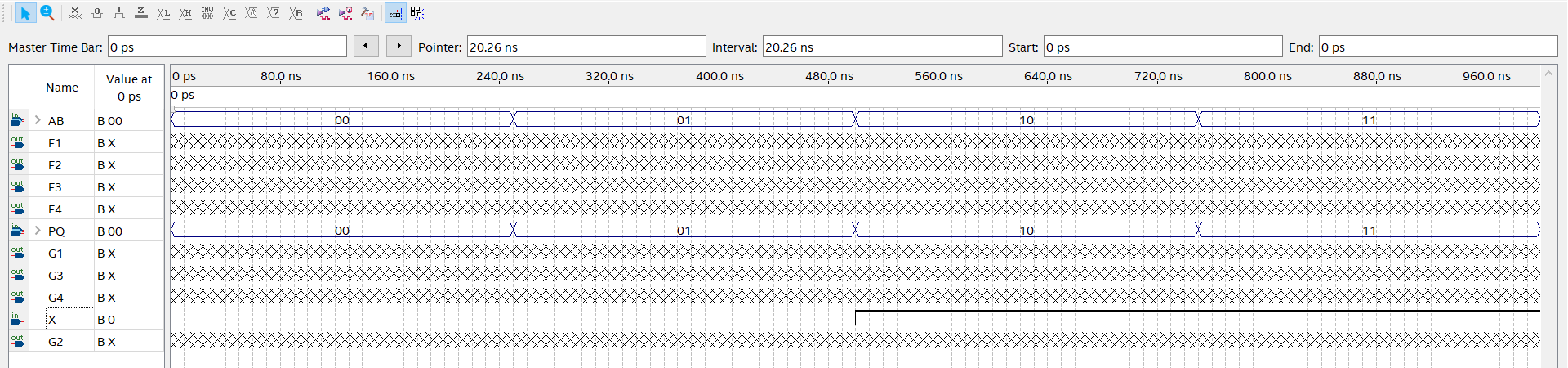


Figure 1.11: Completed Input Values for all signals. Yours might look different once you have made it.

1. Choose **Simulation** > **Run** **Functional Simulation**. Let the simulation complete. At the end, a window will open containing your simulation result.
2. **Doing Your Project**

*You will find the Project Questions on your Canvas site in a Quiz that your instructor will make available during the duration of the Project. You should be able to save your progress and return to the questions during the period that this Project is active. Follow your instructor’s guidance.*

*Part 1: Functional Completeness*

P0Gates contains four circuit diagrams laid out using NAND gates. Each circuit diagram corresponds to a logic gate that you have studied in class. You may find the layout to be strange, since each circuit diagram shows a function that has been created using only NAND gates.

The NAND gate is a functionally complete gate. In the basic sense, this means that we can implement each of the other logic functions using only NAND gates. In this section of the project, you will explore this concept by determining the logic function that each of the four circuits implements.

Part 1 of the Online Project Questions contains a truth table that corresponds to each of the circuits. Start with the first circuit – the one having output F1. Using the results from the simulation you performed in the previous section, record the output value for each input combination in the first truth table. Use the output information to identify the gate implemented in the first circuit.

Repeat these steps for each of the other circuits. Using the simulation results, record the output values for each input combination in the appropriate truth table. Use the output information to identify the gate implemented in the corresponding circuit.

*Part 2: Bubble Propagation*

Even though we can implement each of the gates contained in P0Gates using only NAND gates, it is not particularly efficient to do so using the circuits shown there. Fortunately, we have a way to transform sum-of-products circuits directly into a form that uses only NAND gates.

P0Bubble contains four circuit diagrams. The first circuit (having output G1) shows a sum-of-products circuit. Using the simulation results, record the output values for each input combination in the first truth table. Use the output information to identify the gate implemented in the first circuit.

The second circuit (having output G2) shows a “double-inverter” circuit. Input signal *X* is the input to an inverter whose output is the input to a second inverter. Thus, the output of the second inverter is a doubly-complemented version of input signal *X*. Compare the input and output values in the simulation. *Does inverting a signal twice change its value? Which Boolean algebra principle expresses this fact?*

The third circuit (having output G3) shows a modified version of the first circuit. Think of the bubbles as inverters. The AND gate outputs have been inverted, and so have the corresponding OR gate inputs. Using the simulation results, record the output values for each input combination in the third truth table. Use the output information to identify the gate implemented in the third circuit.

In the third circuit, it should be obvious that the two first-level AND gates have been changed into NAND gates. But what about the second level OR gate? *What kind of gate does an OR gate become if we invert all of its inputs? Which of your Boolean algebra principles expresses this fact?*

The fourth circuit (having output G4) represents the final result of bubble propagation. Compare the outputs of the third and fourth circuits. *Does changing the invert-OR gate into its equivalent change the function’s output behavior?*

Bubble propagation is a simple way to transform circuits into a form that uses only one kind of gate. You must be careful when using bubble propagation, as it only works for circuits that are organized in specific ways. You will learn more about bubble propagation from your text and your class lectures.

*Part 3: The XNOR Gate Circuit*

Copy **P0Implementation.qar** to the working folder you created in Step 1 of Part 1, and open the archive as you did with P0Simulation.qar. Doing this should create a new folder containing the files for this part of the Project. After restoring the archived project, double-click P0Implementation in the Entity window. P0Implementation contains a component called XNORGate. The inputs and outputs of this component have received pin names so that you can implement the circuit on your DE0 Nano Board.

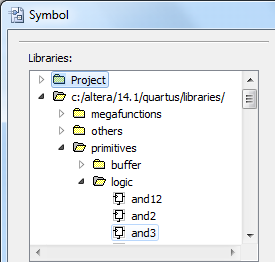
Double-click XNORgate. It should open XNORGate.bdf, which is an empty schematic containing only two input pins and an output pin. Your task is to construct a circuit to implement a 2-input XNOR gate in this schematic.

Figure 3.1 shows the buttons that you can use to edit a BDF file.



Figure 3.1: A portion of the Circuit Design Toolbar

* To select a component, choose the  button and click on the component you want to select.
* To add logic gates to a schematic, press  to open the dialogue box shown in Figure 3.2



Primitive gates

Figure 3.2: Component Selection

* To wire between pins, use , or click on a pin and drag a wire to the point where you want it to end. Be careful, as dragging wires or parts improperly can result in undesired wire connections.

Hint: Be careful you do not use the arrow tool to draw wires between gates and ports. This sometimes generates a wire, but almost always causes connection problems.

Hint: Be sure to use the node tool, not the bus tool to draw your wires. They look similar, but the bus tool has a thicker line. If you hover over the toolbar, the name of the tool will pop up.

You should now be able to add your own logic to XNORGate.bdf. Use your knowledge of how the 2-input XNOR gate can be implemented as a sum-of-products circuit to determine how you can create the XNOR gate *using only 2-input NAND gates and inverters*. ***Your circuit may contain a maximum of three 2-input NAND gates and two inverters*.** ***Do not use an XNOR gate to represent the XNOR gate!*** Use the layout of the circuits in P0Bubble as a guide for creating organizing your XNOR gate circuit. Use what you have learned about simulation to verify that the circuit you have designed has a behavior matching that of the XNOR function.

After you make changes to a schematic, generate a new symbol file by choosing **File > Create/Update > Create Symbol Files for Current File.**

After you have verified the workings of your circuit, you can use Quartus to compile the design and implement the circuit on your DE0 Nano Board. To compile your design, double-click **Compile Design** in the Tasks window, or select **Processing > Start Compilation** from the toolbar. There are certain warnings that your implementation might generate during compilation that you can safely ignore, but if you receive errors or critical warnings or errors, you should correct them before continuing. When the compilation completes with no errors, it will generate a .sof (SRAM Object File) file that you will use to program the FPGA on the DE0 Nano board. The .sof file will have the same name as the project.

To program the DE0 Nano board, connect the board to your computer using the USB cable provided with the board. Double-click **Program Device (Open Programmer)** in the Tasks window, or select **Tools > Programmer** from the toolbar. The programmer window should open, as shown in Figure 3.3. The .sof file associated with the current project should appear as the file to be downloaded to the board. If it does not appear, select “Add File” and then select the .sof file from the project directory.

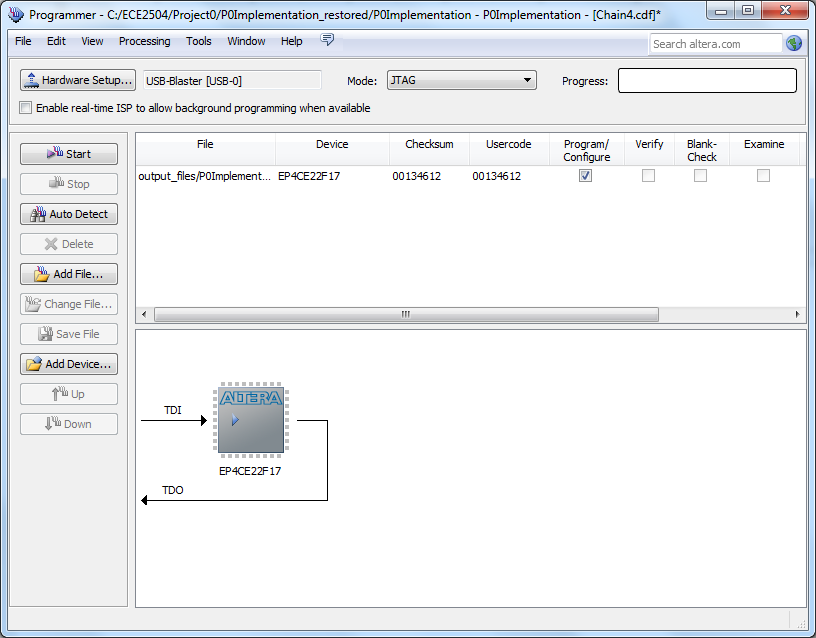


Figure 3.3. Sample Programmer Window.

**USB-Blaster [USB-0]** should appear in the box beside the **Hardware Setup** button. If it doesn’t, then click **Hardware Setup**, and choose **USB-Blaster [USB-0]** on the **Currently selected hardware** pull-down menu. *If the USB-Blaster option does not appear, obtain help from your instructor or a GTA with the installation of your drivers.* After selecting the correct hardware option, select **Close** to return to the Programmer window.

In the Programmer window, click on the **Start** button. The Start button will gray out, and the LEDs on the DE0 Nano board will dim. After a few seconds, the Progress indicator on the programmer window should turn green and say 100% (Successful). Your design has now been downloaded to the board.

To test your design, flip DIP switches SW1 and SW0 and observe the value of LED0. With the board positioned so that the LEDs are at the top of the board, SW1 and SW0 are the two switches on the right of the bank of DIP switches. A switch provides logic-1 when it is in the up position and logic-0 when it is in the down position. Operate your circuit by trying all four combinations of the two input switches, and check the value of the LED output to see if it matches the desired behavior of an XNOR gate.

1. **Completing Your Project**

On or before the due date and time, you must complete the Online Project Questions using Canvas or Scholar.

This project does not require CEL validation. Instead, you will provide the source files that will allow the GTA to compile the same files that you used to implement your XNOR gate circuit on the DE0 Nano Board. Create an archive of your work in **P0Implementation.bdf** by choosing **Project > Archive Project** after you complete the implementation. When prompted for a name for your archive, the default archive name will be the same original archive. *Append your Virginia Tech PID to the end of the filename.*

When you create the Quartus archive, it should appear in the same folder that was created when you opened the original archive. Upload the archive to Canvas or Scholar, making certain that you upload the completed archive that you created, and not the one that was provided to you. The file that you submit is the file that will be graded.

Upload a blank copy of the included validation sheet with your project submission. *Append your Virginia Tech PID to the end of the filename.*

Your instructor may require additional elements. Follow your instructor’s guidance.