

Introduction to Quartus CAD

A tutorial example in design

In this lab, you will be replicating essentially the same lab done by ECE2031 students in their first week of lab. You're not really behind them – they do this tutorial first because it's easy to do even when students have not acquired all of their lab parts. You will not be doing the same prototyping exercises that they do in Labs 2/3 and 4/5 because you have more hardware experiences later in ECE2883, so you will quickly move through combinational and sequential lab exercises in the next couple of weeks.

There is one major difference in Quartus for you, relative to ECE2031 students.

- They use a version of Quartus (9.1 SP2) which is better for learning simulation, so ECE2031 has never advanced beyond it, even though it is now up to version 15.
- You will need version 13.0 SP1, because it is the best compromise between ease of simulation and support for the FPGA used on ECE2883HP's development board.

So, if you install a version of Quartus on your own computer, I recommend that you install 13.0 SP1, and you can wait at least a week before doing it. You are better off to use lab computers, at least during this tutorial lab, and follow the tutorial instructions for using version 9.1 SP2. You will get lost if you try to follow the tutorial instructions on a computer that has any version newer than version 9.1 SP2.

I have already written a new version of the tutorial for 13.0, but I would prefer that you not be test subjects just yet.

So, with that lengthy introduction, continue by following the steps that begin on the next page. Note that there are two PDF comments telling you to save two simulation results to turn in. Those two items are the only items to turn in, and you can either give them to me or Kevin in lab, or bring them to seminar on Thursday.

Appendix B: Quartus II Tutorial

The purpose of this tutorial is to familiarize the student with the basic functionality of the Quartus II software. The concepts and procedures outlined here should enable the student to begin working on laboratory projects. Additional features will also be explained throughout the laboratory exercises. However, the student should refer back to this tutorial if at any time he or she forgets any of these basic functions.

This tutorial has been written using Quartus II 9.0, but it should be applicable for any later revisions as well. The exact menus, display windows, and other options may change slightly between versions, but the basic functionality should remain fairly constant.

B.1 Creating a new project

Upon starting Quartus, the main interface window will be presented as seen in Figure B.1. From the menu at the top, select **File => New Project Wizard...** to create a new project. At this point, Quartus may display an introduction screen for the project wizard. Simply press **Next >** to advance

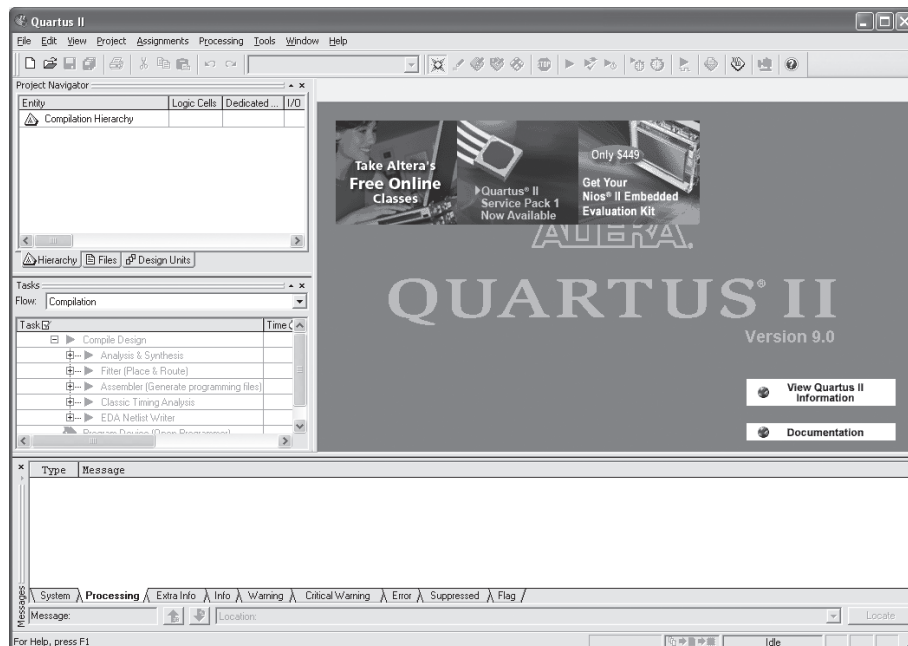


Figure B.1. The Quartus II development environment.

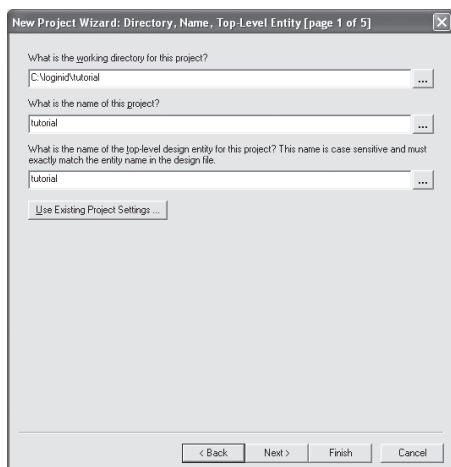


Figure B.2. The first dialog of the New Project Wizard (possibly after an introductory screen).

to distinguish your files from other students who may use that machine. Finally, append a directory for the particular project, which is *tutorial* in the example of Figure B.2. Next, give the project a name. The example uses *tutorial* again, but this name does not have to be the same as the project directory. The top-level design entity can also be something other than the project name or directory, but for simplicity, Quartus II will default to the project name. Press **Next >** to advance to the window in Figure B.3 (left). Allow Quartus to create the project directory, if it does not yet exist.

Note!

Quartus has a multi-document organization, allowing different documents/applications to show up either as sub-windows or tabs. These windows or tabs can be closed independently of one another, and reopened when necessary. When closing one of them, Quartus will warn of any unsaved changes relative to that item. One such sub-window is the “Tips and Tricks” that will pop up until you change your preferences otherwise.

Since there are no design files to add to this project, click **Next >** to advance to the device selection window of Figure B.3 (right). You may be using either the DE2 board or the DE2-70 board. These boards contain different FPGA devices. The DE2 board uses the EP2C35F672C6 FPGA, while the DE2-70 uses the EP2C70F896C6 (at the time this was printed, at least). You will need to select the correct device for the board at your laboratory

to the window displayed in Figure B.2. You may also want to check the box in the lower left corner to avoid displaying the introduction screen again.

Enter a location to store your project files. You should **not** use a network drive, because you may experience slow or problematic compilations. If you use the local hard disk on one of the machines in the laboratory, choose a temporary location, such as *c:\work* (acceptable locations may change based upon the policies of computer support personnel). You may also want to use your user name (login ID)

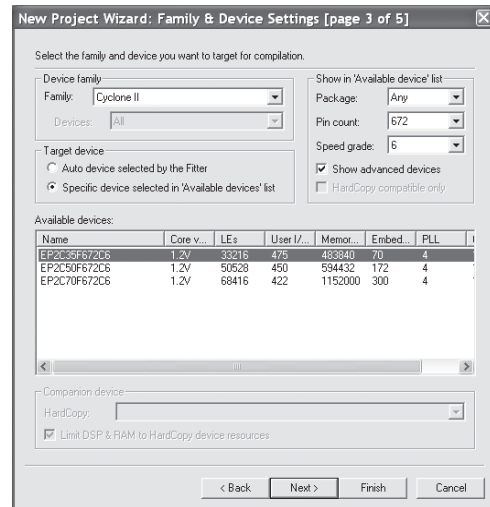
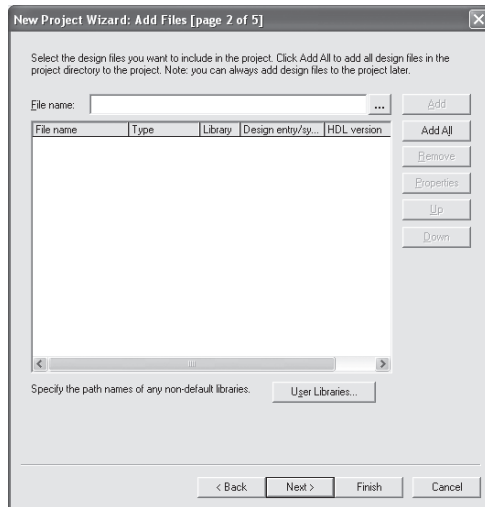


Figure B.3. The second dialog allows a user to add previously created design files to the new project. The third dialog specifies the target hardware for the project.

station by reading the corresponding code near the middle of the FPGA IC (*Integrated Circuit*). If you change laboratory stations, you may need to update the device information to the alternate FPGA and re-compile the source files before downloading to the FPGA board. If you are completing this part of the tutorial with no hardware (e.g., at home), assume the EP2C35F672C6 FPGA.

By default, Quartus shows all FPGAs of a particular family currently available from Altera. Change the **Family** selection to **Cyclone II** (for DE2 boards) to bring up the appropriate family of devices. To select the correct device, you could scroll through the list to find it. However, it is often easier to narrow your search first. The last number in an Altera FPGA designation indicates the speed grade of the device. Select a **Speed grade** according to your device (the last digit of its part number), as seen in Figure B.3 (right). The number just before this speed designation indicates the number of pins that the FPGA has, so change the **Pin count** option to reflect

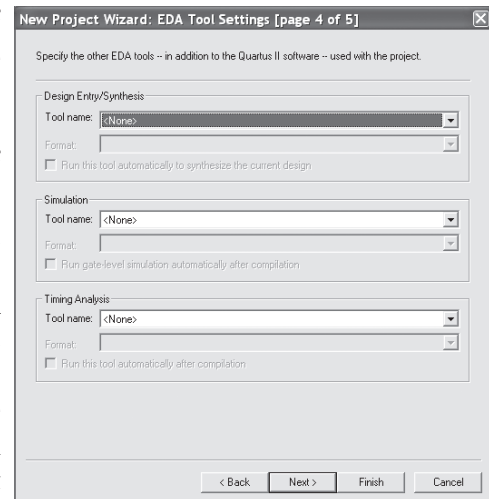


Figure B.4. Optional third-party tools are not used in this tutorial.

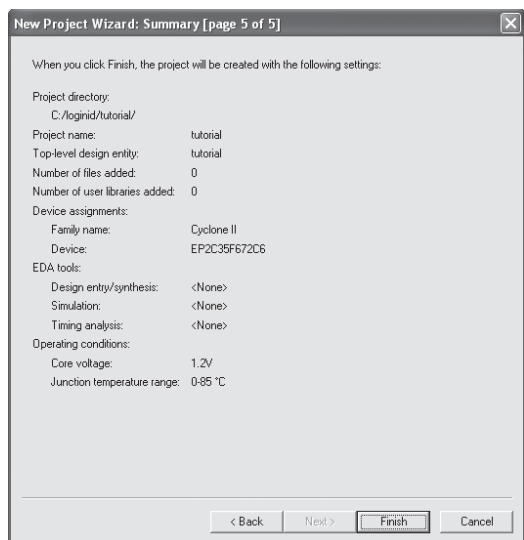


Figure B.5. The final window of the Project Wizard allows the user to see all project parameters.

the **672** (DE2) or **896** (DE2-70) pins on these devices. This should reduce the device options to only a handful of FPGAs. Select the correct device for your laboratory station. Press **Next >** to continue to the window of Figure B.4.

It is common for commercial entities to use third-party tools to develop hardware and software configurations for FPGAs. This window allows Quartus to communicate and use such tools within the development environment. However, we will simply be using the default tools included with Quartus II, so click **Next >** to proceed to the summary window of Figure B.5.

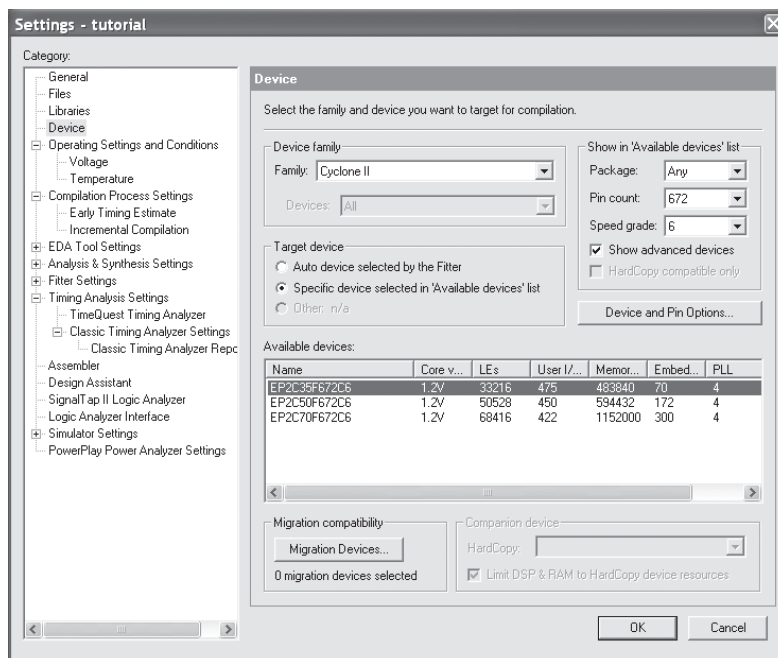


Figure B.6. The Device window can be used to subsequently change the selected device, but is more often used to alter other specific parameters related to the device and its pins.

With the summary window in view, press **Finish** to close the wizard and create the new project files. You will be returned to the main window of Quartus. From the menu, select **Assignments => Device...** to bring up the settings window of Figure B.6.

Click on the **Device & Pin Options...** button and select the **Unused Pins** tab to bring up the options window of Figure B.7. Change the **Reserve all unused pins:** option to **As input tri-stated**. This option forces all of the unused pins on the FPGA to be inputs instead of the default outputs. In normal new design situations, this would not matter, since the board design would typically be done after the FPGA is designed.

However, the projects in these laboratory exercises will target a prebuilt FPGA board, which has several other ICs and connectors on it in addition to the FPGA. Many of these ICs connect directly to the FPGA I/O pins. To avoid causing erroneous board operation or potential damage to the FPGA or other ICs, it is important to change this setting every time a new project is created. Once complete, press the **OK** button to close this window. Press the **OK** button on the settings window to complete the options change.

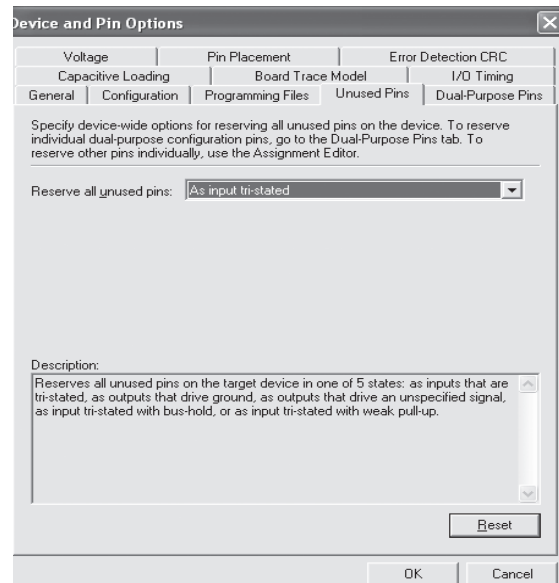


Figure B.7. The state of unused pins on the FPGA **must** be specified to avoid hardware conflicts.

Remember!

Always set the unused pins option to tri-stated inputs whenever you create a new project for downloading to a DE2 board in the laboratory. Failure to do so could result in damage to the DE2 board components attached to the FPGA. Think of it this way: If the compiler happens to drive one of these pins (that it thinks is unused) as an output, and if some device like a USB interface is driving it externally, then either the FPGA or the USB interface will “lose” and possibly burn out.

B.2 Creating a new design file

Design projects are composed of one or more design files, and this project will be defined by a single schematic file. From the menu, select **File => New...** to bring up the dialog of Figure B.8.

Select the **Block Diagram/Schematic File** option and press the **OK** button. This will create a blank schematic file, as seen in Figure B.9. It may default to a tabbed document, as seen in Figure B.10. Either mode is fine, but if you want to have a dedicated sub-window for your block diagram, you can click on the maximize/minimize icon just to the left of the small “X” in the upper right corner. (If you accidentally “X” it out and delete the window, just go back to the step above, and create a NEW blank block diagram.

Look ahead to Figure B.14 if you want to see what we are drawing. Start by double-clicking anywhere in the blank area within the schematic window. This will bring up the symbol insertion window of Figure B.11.

Select the **and2** symbol from the *logic* folder of the *primitives* library. Press the **OK** button and click within the schematic window to place the AND gate. You can move any symbol by simply clicking and dragging. Double-click a clear area of the schematic window to bring up the symbol window again. This time, select the **not** symbol from the *primitives/logic* library.

Check the “repeat-insert mode” and press the “OK” button to place multiple NOT gates. Place two NOT gates and press the escape key (ESC) to quit placing symbols. Right click on one of the NOT gates and select **Rotate by Degrees => 90**. Rotate the second NOT gate by 270 degrees. In a similar manner, add the components shown in Figure B.12.

The **or2** is also located within the *logic* folder of the *primitives* library, but the **input** and **output** symbols are located within the *pin* folder of the same *primitives* library. If you already know what the symbol name is, you can also type it into the name section of the symbol window

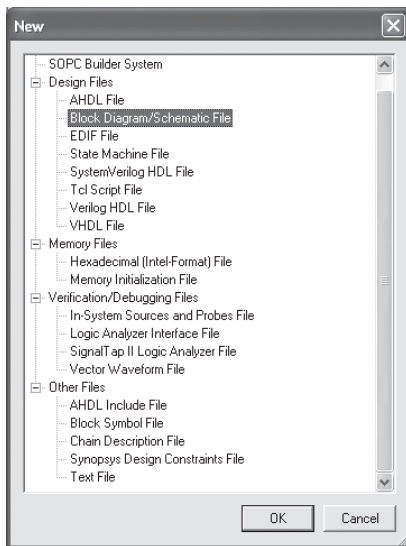


Figure B.8. Block diagrams and other design files are created from this window.

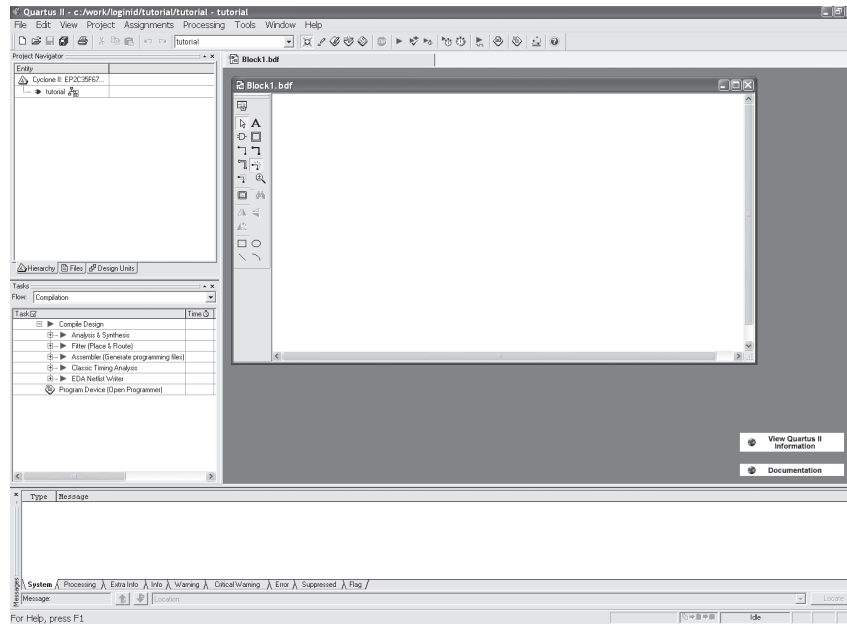


Figure B.9. A blank schematic, or “block diagram.”

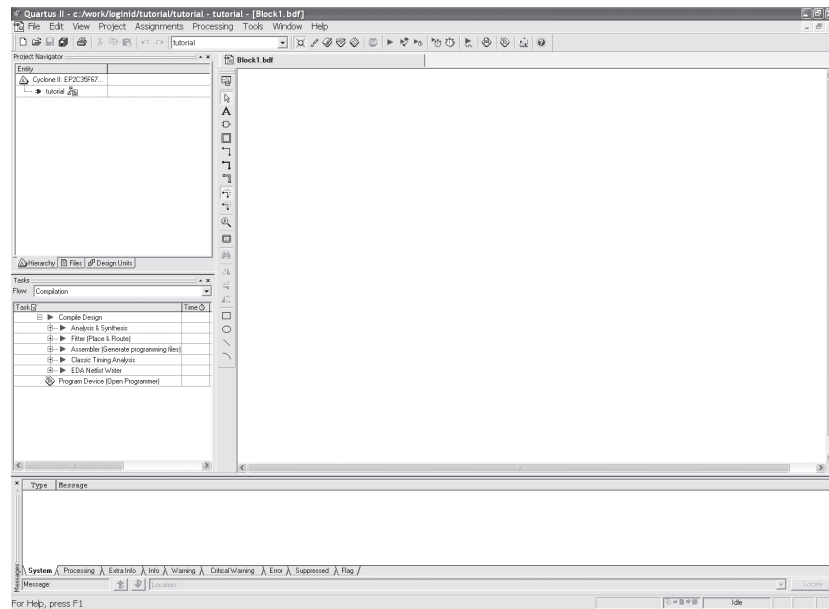


Figure B.10. The same block diagram window as a tabbed document.

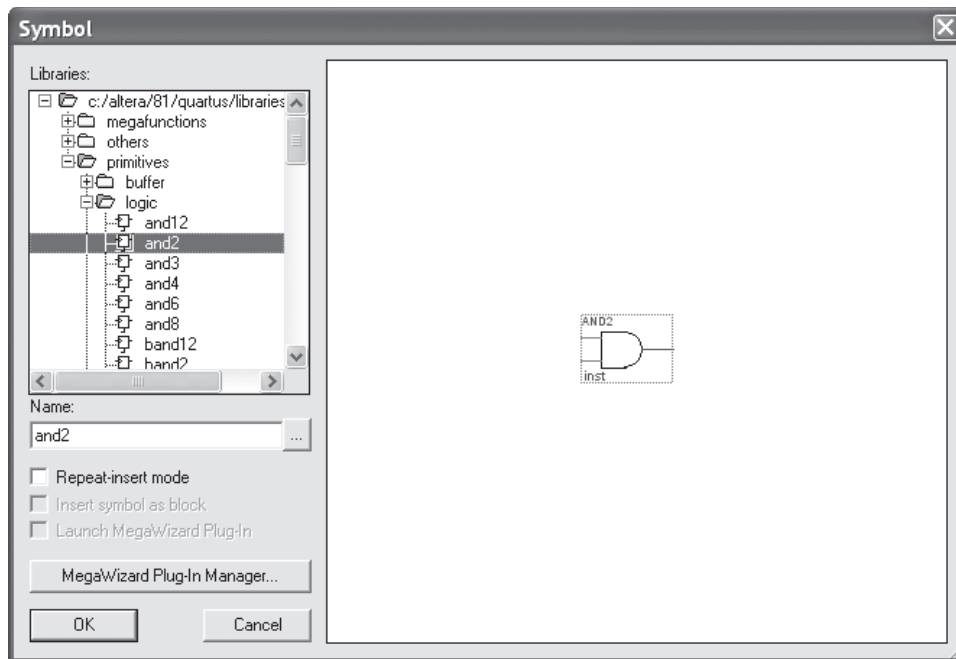


Figure B.11. Symbols can be inserted from any of several libraries.

instead of selecting it from the library tree on the left. Arrange the symbols according to Figure B.12. Double click on the top left input pin to bring up the pin properties, seen in Figure B.13.

Change the “Pin name” to **sw4** and press the “OK” button. Likewise, change the name of the second input to **sw5**. An alternate way to change the pin name is to first select the pin and then double click on the pin name text. Try this on the top right pin and change the name to **d6**. When you are done, press enter and the next pin name below d6 will be highlighted.

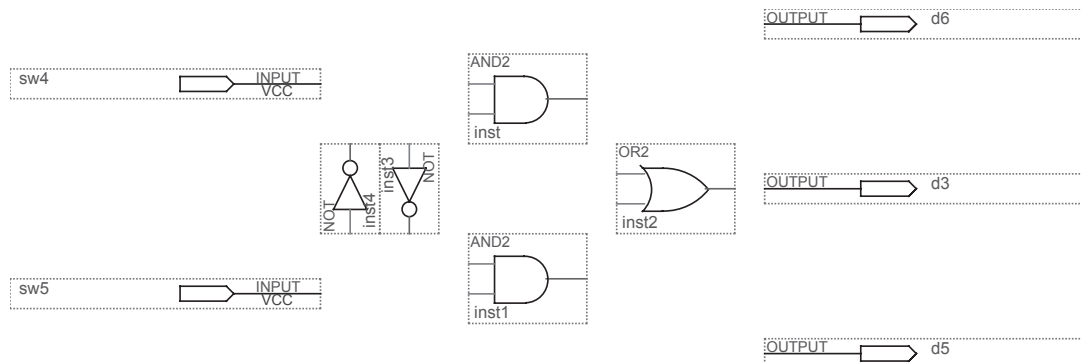


Figure B.12. An assortment of unconnected symbols.

This feature will allow you to name a series of pins without pulling up a properties window. Name the bottom two outputs d3 and d5 according to Figure B.14.

Now add the wires between each of the symbols within the schematic. One way to do this is to use the **Orthogonal Node Tool** located on the left hand toolbar. Another way is to simply move the mouse over one of the symbol terminals, which causes the mouse to change to a crosshair and the icon for the orthogonal node tool. Click and drag to draw a wire from the terminal to another symbol's terminal and release the mouse button. You can also draw multiple wire segments by releasing the mouse button over a blank area of the schematic. If you release the mouse button while dragging a wire over another wire, a connection between the wires will be made in the form of a solder dot. Using these techniques, make all of the connections shown in Figure B.14. If you ever need to move devices around, select the arrow tool to get out of the orthogonal node tool mode

The next step is to verify that there are no errors or serious warnings within the design file. First save the file using **File => Save As...** and use the default filename, which should be the top-level entity name that you chose when creating the project, with a .bdf extension. Leave the box

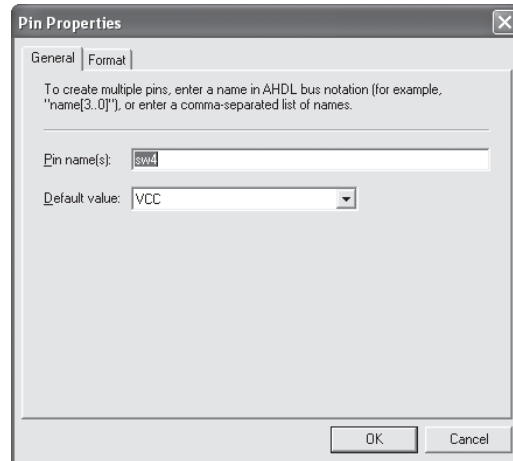


Figure B.13. Pins must be given meaningful names to make the design understandable.

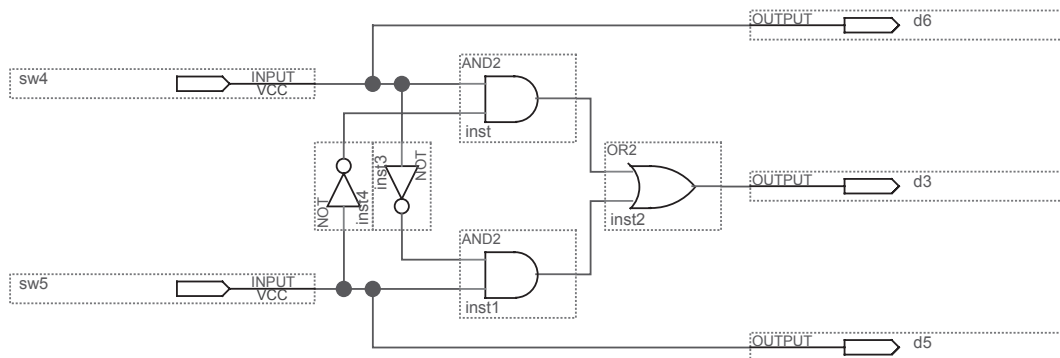


Figure B.14. A connected schematic with pin names.

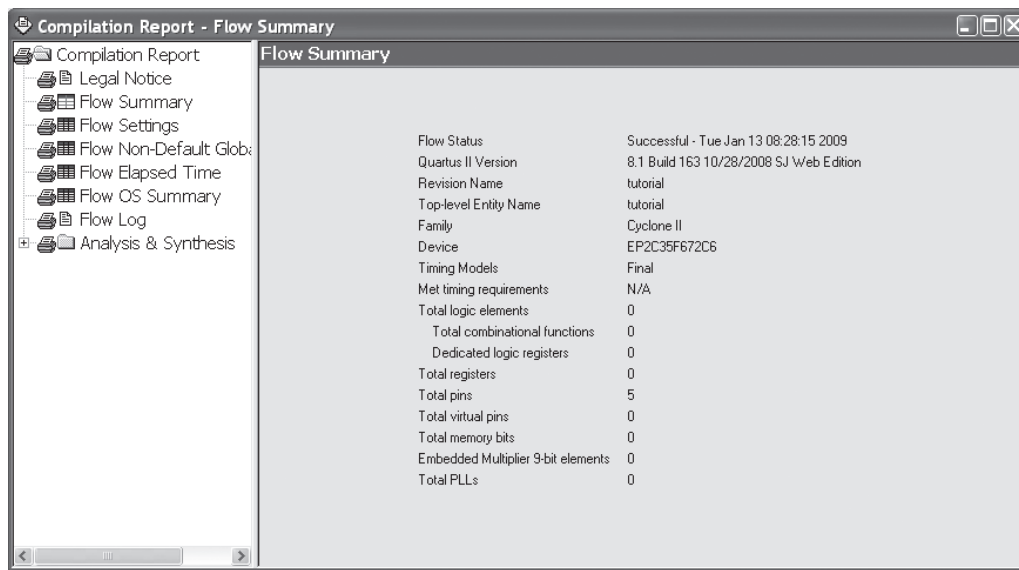


Figure B.15. The result of successful analysis and synthesis (partial compilation).

checked that indicates that this design file will be added to the current project. (You could always add it to the project later as a separate step, working from the “Files” tab of the “Project Navigator” pane.)

With the file saved, compile the design by selecting **Processing => Start Compilation**, or by clicking on the corresponding icon in the top toolbar (a right-pointing triangle that looks like a video “play” button). If everything has been connected correctly, there should be no errors or warnings and the Compilation Report will be displayed, as seen in Figure B.15. You could also make a partial compilation with **Processing => Start => Start Analysis and Synthesis**, but you can simplify the Quartus learning process by always compiling every project completely. Still, analysis and synthesis by itself would be slightly faster and actually would be sufficient to simplify the next step of adding pins assignments.

Table B.1 Pin assignments		
Node name	DE2 pin	DE2-70 pin
d3	PIN_AC22 (LEDR[3])	PIN_AJ4 (LEDR[3])
d5	PIN_AD23 (LEDR[5])	PIN_AH4 (LEDR[5])
d6	PIN_AD21 (LEDR[6])	PIN_AJ3 (LEDR[6])
sw4	PIN_AF14 (SW[4])	AF14 (SW[4])
sw5	PIN_AD13 (SW[5])	AD13 (SW[5])

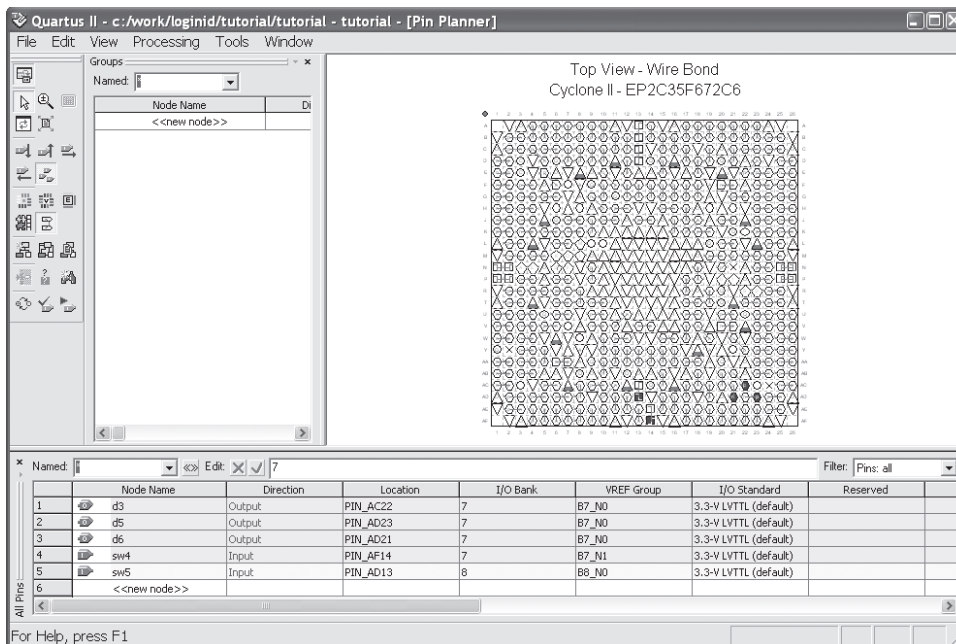


Figure B.16. The Pin Planner is a convenient way to assign specific FPGA pins to pin symbols in a design.

The final step in the design process is to add pin assignments to the design file. Open the pin planner by selecting **Assignments => Pin Planner**, which brings up the window in Figure B.16.

Notice that the I/O pins are listed in the table at the bottom of the Pin Planner. During the analysis and synthesis step, a list of I/O pins was made by Quartus, which is displayed here. Adding pin constraints before performing a synthesis or full compilation step would require manually entering each I/O pin in the list. For each I/O pin in the list, double-click on the **Location** cell and select the pin numbers shown in Table B.1. Refer to Chapter 1 or Appendix C of the lab manual (or the DE2 manufacturer's reference manual) to understand why these particular pins were used (they correspond to hardwired connections to LEDs and switches on the DE2 board.)

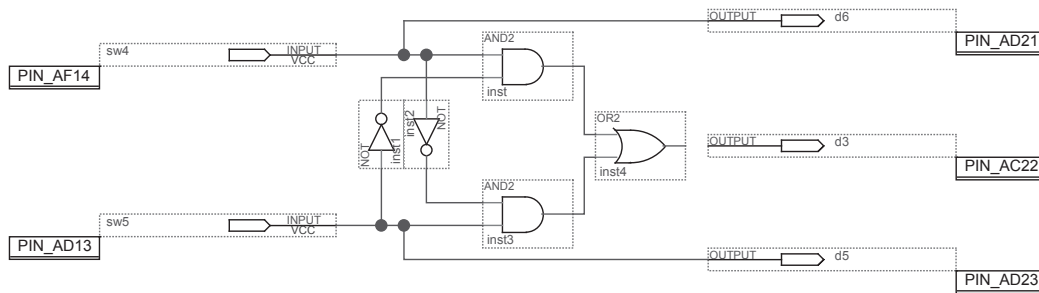


Figure B.17. A completed block diagram with pin assignments (DE2 values used here).

Remember!

There are two kinds of signal pins on the FPGA: **Unused** ones (that we specified earlier as always being “input, tri-stated,” to avoid problems), and **used** ones, which we always have to assign to their **CORRECT** location on the DE2 board. It doesn’t make sense to use a pin without knowing what is actually connected to it (a switch, an LED, or whatever).

With all of the pin assignments made, close the Pin Planner window. The pin numbers should now appear beside the I/O pins, as seen in Figure B.17. (If they do not show up, select **View => Show Location Assignments** from the Quartus II menu. Sometimes, it seems to be necessary to turn this view option OFF, then back ON again.)

Hint!

By default in the block diagram editor, Quartus places the pin assignments to the side and slightly below the pin, exactly as shown in Figure B.17. This can get confusing when pins are stacked one above the other, because it is less obvious which assignment belongs to which pin. You may want to drag the pin assignments directly alongside the pins, or even off further to the side (a small line will appear to show the association between pin and assignment).

B.3 Simulating the design

The first step in simulating the design is to perform a full compilation. From the menu, select **Processing => Start Compilation** or click on the triangle icon in the top toolbar to start a compilation. Once complete,

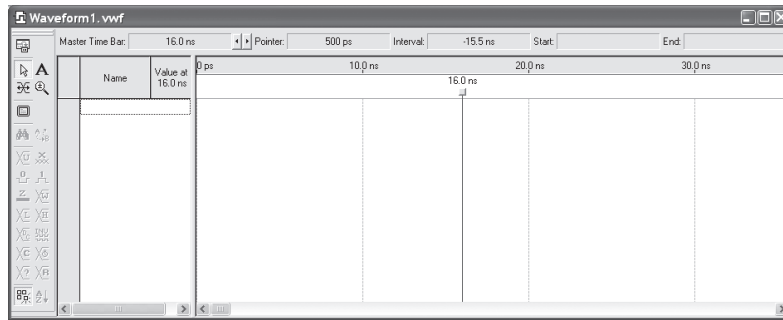


Figure B.18. A new blank waveform file.

the compilation report will display a summary. Press the **OK** button to view the compilation report when the message box appears telling you the compilation is complete.

Close the compilation report and create a new waveform file by selecting **File => New...** from the menu. Under the *Verification/Debugging Files*, select **Vector Waveform File**, and press the **OK** button. This will create a blank simulation waveform file as seen in Figure B.18. Right click on the left hand side of the window (in the blank area under “Name”) and select the **Insert => Insert Node or Bus...** option to bring up the window in Figure B.19. A *node* is a signal, and a *bus* is a group of related signals. They may or may not be assigned to specific pins.

Click on the **Node Finder...** button to bring up a window similar to Figure B.20. For this simulation, we are only interested in the input and output pins, although it is possible to show internal signals in the simulator. So first select only “Pins: assigned” in the Filter option, to reduce the number of signals that will be shown in the Node Finder window. Then click on the **List** button to search for assigned pins and to fill in the **Nodes Found** list. (If the pins had not already been assigned a location, as we did with the Pin Planner earlier, you would have needed to broaden the filter to include “All pins” instead.) Now that the list is populated, highlight all of the pins (Ctrl-Click) in the

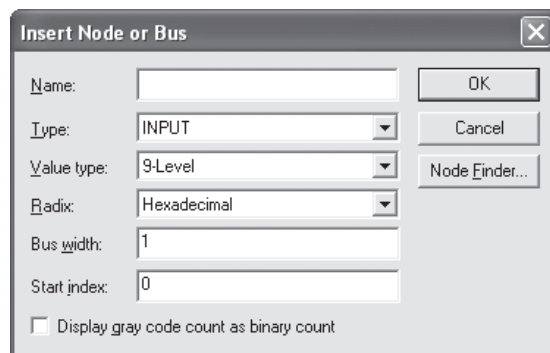


Figure B.19. Specific signal names can be entered to add them to the waveform, but this window is usually used to access the more convenient “Node Finder.”

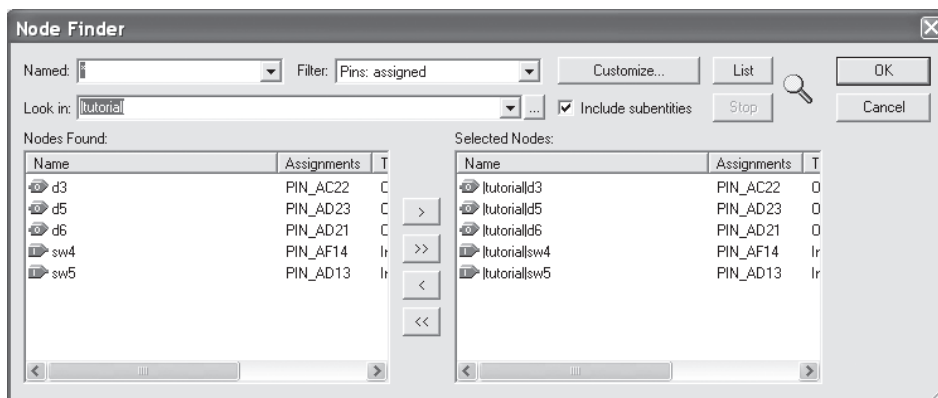


Figure B.20. Pressing the “List” button in the Node Finder generates a list of all signal nodes satisfying the filter setting, and desired nodes can be selected with the right arrow buttons. The double right arrow selects ALL found nodes, and can also be used in this example, since we would like to show all assigned pins in the simulation.

Nodes Found list and click on the > button to select them (moving them to the “Selected Nodes” group on the right. You should now have a window that looks similar to that of Figure B.21. It will show “Multiple Items,” and you should change the radix to “Binary” if it is set otherwise (as in the figure, where it defaulted to “Hexadecimal.”

Press the **OK** button to update the Insert Node or Bus dialog. Press **OK** on this window to insert the pin nodes into the waveform file. You should now see the list of pin names in the left panel of the waveform window. You can click on the individual signals and then drag them to change the order. From the menu, select **Edit => End Time...** and enter a

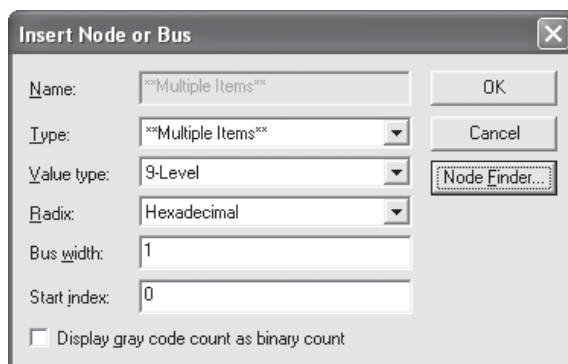


Figure B.21. Completion of the Node Finder returns to the Insert Node or Bus dialog.

value of 200 ns, as seen in Figure B.22. A value of 200 ns is chosen here only because it has been predetermined that 200 ns will be long enough to allow each of the four permutations of sw4 and sw5: (0,0), (0,1), (1,0), and (1,1).

Press the **OK** button. Right click on the right panel of the waveform window and select **Zoom => Fit in Window**. This will allow you to see the entire waveform. Next, select the sw4

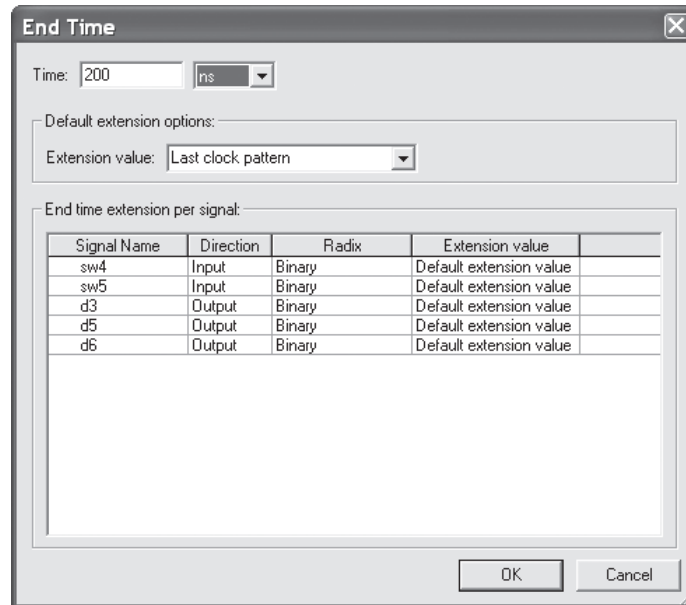


Figure B.22. The simulation will only run to the end time specified by the user.

signal and right click on it. Select **Value => Clock...** to bring up the clock window of Figure B.23.

Technically, neither of the two inputs of this circuit are “clocks,” since this is a simple combinational logic circuit. Still, the **Value => Clock...** option gives a very convenient way of toggling an input at specified intervals. For this example, we want sw4 to toggle once at the middle of the simulation, and we want sw5 to toggle three times (look ahead to Figure B.24 to see where this is headed). The “clock” should continue for the duration of the simulation, which is 200 ns in this case, so set the “End Time” to 200 ns. Change the “Period” to be 200 ns to make it toggle just once in 200 ns, and press the **OK** button to update the sw4 signal.

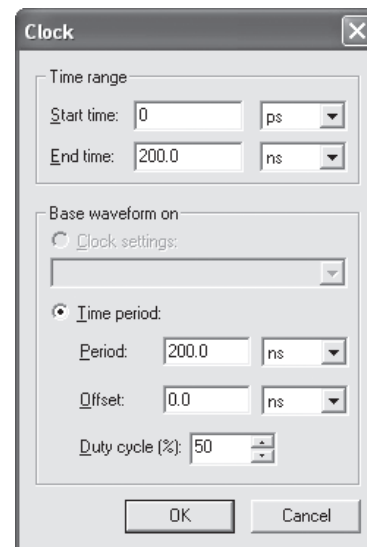


Figure B.23. The clock window allows an input signal to be given the value of a square wave.

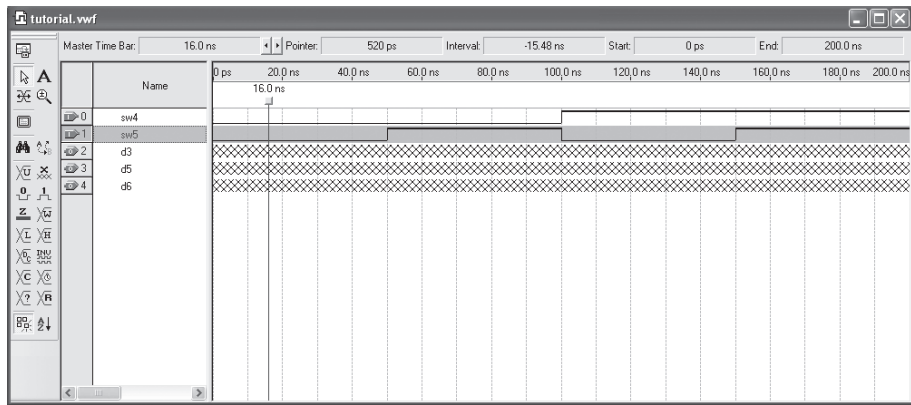


Figure B.24. The sw4 signal is created as one clock period over the whole simulation duration, while the sw5 signal cycles through two periods, producing every possible combination of highs and lows for the two signals.

Now, to make sw5 toggle twice as fast, change the value of the sw5 waveform to be a clock with a period of 100 ns, as seen in Figure B.24. Alternately, you could select individual sections of the waveform using the mouse and change those segments by right clicking on the highlighted segment and selecting the **Value** menu items (try it). Now save the waveform file

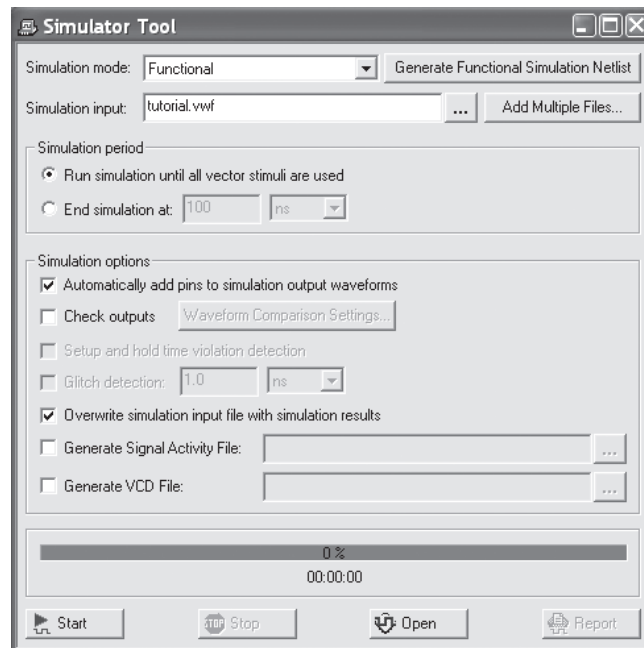


Figure B.25. The Simulator Tool window defines the type and parameters of a simulation run.

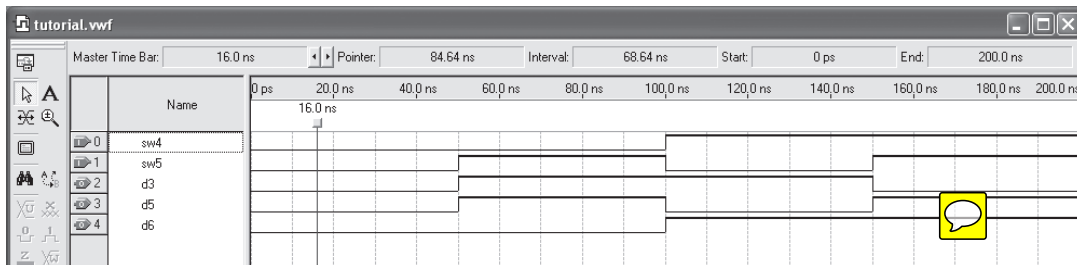


Figure B.26. The result of a functional simulation.

using **File => Save As...** dialog and using the default filename (creating *tutorial.vwf* in the project directory). From the menu, select **Processing => Simulator Tool** to open the window of Figure B.25.

Change the **Simulation mode** to **Functional** and press the **Generate Functional Simulation Netlist** button. Once complete, a dialog box will inform you that the generation was successful. Press the **OK** button to return to the simulator tool. Make sure the “Overwrite simulation input file with simulation results” box is checked and press the **Start** button. Once complete, a dialog box will be displayed to tell you so. Press the **OK** button to return to the simulator tool once again. Press the **Open** button to display the results of the simulation. If you left the waveform window open in Quartus, it will prompt you to reload the waveform file. Select **Yes** to continue and display the waveform window of Figure B.26.

Notice that the pins d5 and d6 match the waveforms for sw5 and sw4, respectively. Since we tied these pins together, that makes sense. The output of the circuit, d3, is the exclusive-or of the two inputs, which is also as expected. The functional simulation informs you that the combinational

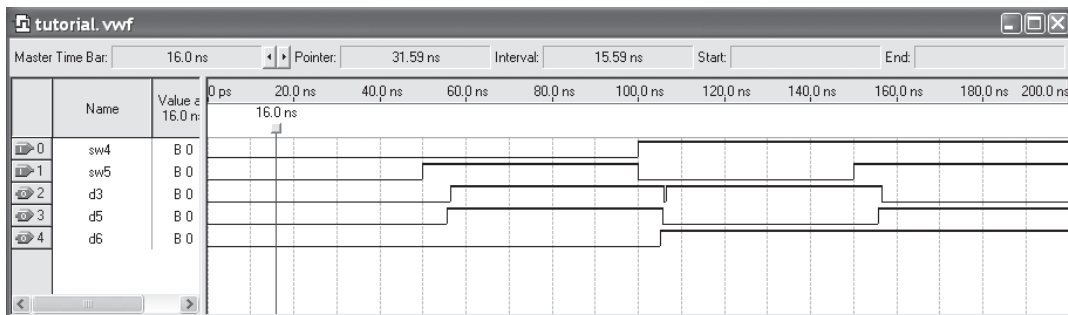



Figure B.27. The result of a timing simulation, which accurately reflects actual time delays.

logic generated is performing the desired task. However, this is not a guarantee that the circuit will perform correctly when implemented on the FPGA. To determine this, a timing simulation needs to be performed that will take into account the various signal delays observed within the routing and components of the FPGA.

Return to the simulator tool window and change the **Simulation mode** to the **Timing** option and start the simulation. Once complete, open the results, which should be similar to that of Figure B.27. 

Notice that the outputs no longer instantly change at the edges of the inputs. There is a slight propagation delay. There may even be glitches, as shown in Figure B.27. Imagine how this can be caused by some signals propagating to gate inputs sooner than others. This will become especially important when working with sequential circuits later when determining maximum clocking rates.

B.4 Downloading the design to the DE2

Connect your computer to your target development board. This should require only a simple USB cable between the computer and the DE2 board, where it should be connected to the **leftmost** of the two USB connectors at the top left corner of the board. The connector on the right may be covered with something to prevent you from accidentally using it.

Before downloading your design file, open the device setting window by selecting **Assignments => Device...** from the menu. Verify that the

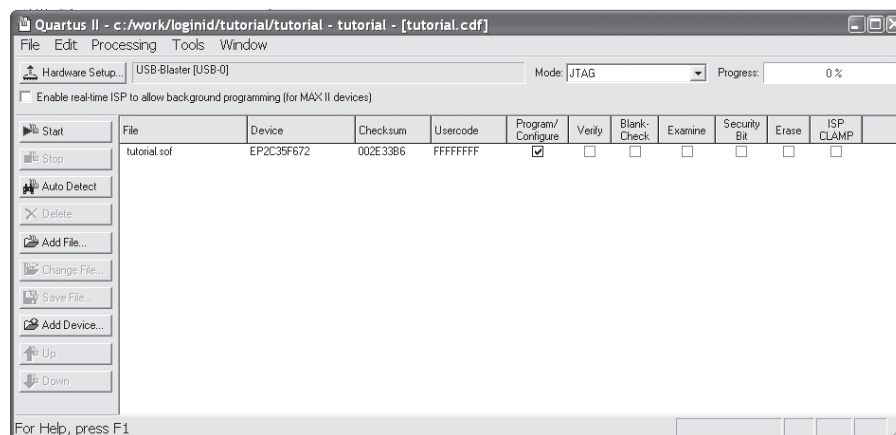


Figure B.28. The Programmer Tool specifies one or more files to be programmed into one or more devices.

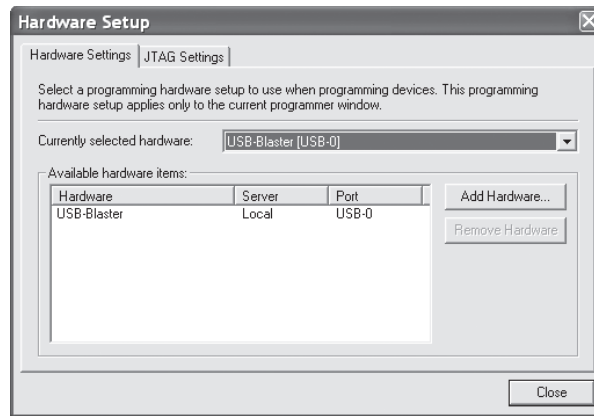


Figure B.29. Programming (or “downloading”) is done with the USB-Blaster (a USB cable connected to the LEFTMOST of the two upper left USB connectors on the DE2).

correct FPGA device is selected. If the wrong device is shown, select the correct FPGA and press the **OK** button. This could require you to go back and reassign pin numbers, but should not be necessary if you completed earlier steps with the correct device assignment.

Open the programmer tool, as seen in Figure B.28, by selecting **Tools => Programmer** from the menu.

Make sure the box under the **Program/Configure** heading is checked, so that the programmer knows that you want to program that file. If the **Hardware Setup** is listed as **No Hardware**, click on the **Hardware Setup...** button to display the window of Figure B.29.

Under the **Currently selected hardware** option, choose your programming hardware and close the window. If no programming hardware is available, make sure that the USB cable is in place and verify that the DE2 board is turned on. With the hardware properly selected, press the **Start** button to begin the download. When complete, the Progress meter in the top right corner will read 100%.

Start with both SW4 and SW5 in the UP position. Notice that the LEDs LEDR6 and LEDR5 are illuminated on the board. Lower the SW4 slide switch. LEDR6 should turn off, and LEDR3 should turn on. Is this the correct operation of the XOR circuit? Take into consideration that the LEDs are active high, meaning that they illuminate when driven with a high signal. The slide switches are low when in the DOWN position. This is something to consider when using these components in future designs. (The active level is given in the corresponding table of Appendix

C.) Lower the SW5 slide switch as well. Try various combinations of the two switches, and verify if the circuit is operating properly.

This concludes the tutorial. Like many of the exercises in this lab manual, there were many steps. If you did not understand WHY you did some of them, consider doing them again. You may find it useful to perform other tutorials, such as those in *Rapid Prototyping of Digital Systems*, or those on the Altera web site. Keep your eye on the big picture.

In this exercise, you

- created a new project in Quartus,
- specified that it would run on the specific FPGA available on the DE2 board,
- defined the project as having a single source file (a schematic),
- drew the schematic,
- assigned input and output pins consistent with the locations of switches and LEDs on the DE2 board,
- simulated the design to verify that it behaves like the exclusive OR circuit that it really is,
- implemented the design on the DE2 board by configuring the internal elements of the FPGA, and
- tested the design.

Note the correlation between these steps and the general process given in the background reading for Lab 1.