ECE 102 - Digital Circuit Logic
Lab - 2

Quartus II Introduction Using Schematic Design

This tutorial presents an introduction to the Quartus® II CAD system. It gives a general overview of a typical CAD flow for designing circuits that are implemented by using FPGA devices, and shows how this flow is realized in the Quartus II software. The design process is illustrated by giving step-by-step instructions for using the Quartus II software to implement a very simple circuit in an Altera FPGA device.

The Quartus II system includes full support for all of the popular methods of entering a description of the desired circuit into a CAD system. This tutorial makes use of the schematic design entry method, in which the user draws a graphical diagram of the circuit. Two other versions of this tutorial are also available, which use the Verilog and VHDL hardware description languages, respectively.

The last step in the design process involves configuring the designed circuit in an actual FPGA device. To show how this is done, it is assumed that the user has access to the Altera DE2 Development and Education board connected to a computer that has Quartus II software installed. A reader who does not have access to the DE2 board will still find the tutorial useful to learn how the FPGA programming and configuration task is performed.

The screen captures in the tutorial were obtained using the Quartus II version 5.0; if other versions of the software are used, some of the images may be slightly different.

Contents:

Note:
For this lab you may skip the sections which are highlighed. Those sections will provide useful information for subsequent labs.

Computer Aided Design (CAD) software makes it easy to implement a desired logic circuit by using a programmable logic device, such as a field-programmable gate array (FPGA) chip. A typical FPGA CAD flow is illustrated in Figure 1.

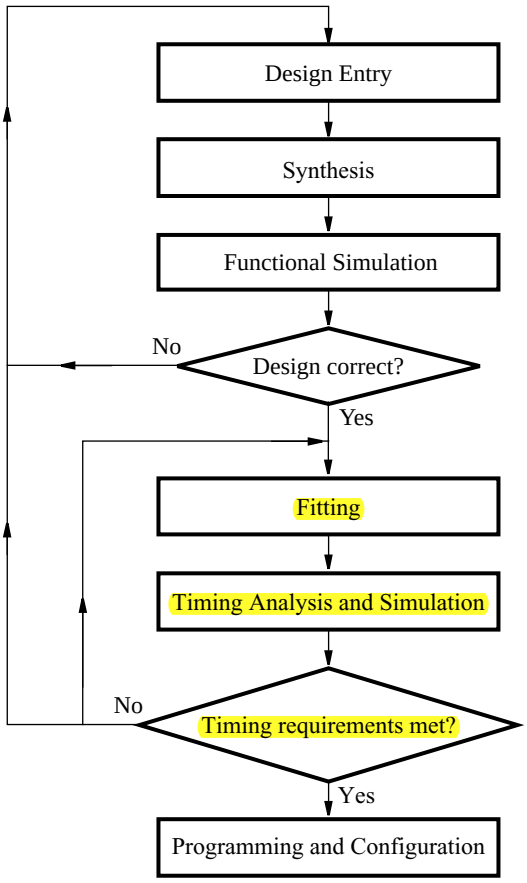


Figure 1. Typical CAD flow.

The CAD flow involves the following steps:

- **Design Entry** – the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as Verilog or VHDL.
- **Synthesis** – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip
- **Functional Simulation** – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues

- **Fitting** – the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs
- **Timing Analysis** – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit
- **Timing Simulation** – the fitted circuit is tested to verify both its functional correctness and timing
- **Programming and Configuration** – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections

This tutorial introduces the basic features of the Quartus II software. It shows how the software can be used to design and implement a circuit specified by means of a schematic diagram. It makes use of the graphical user interface to invoke the Quartus II commands. Doing this tutorial, the reader will learn about:

- Creating a project
- Entering a schematic diagram
- Synthesizing a circuit from the schematic diagram
- Fitting a synthesized circuit into an Altera FPGA
- Assigning the circuit inputs and outputs to specific pins on the FPGA
- Simulating the designed circuit
- Programming and configuring the FPGA chip on Altera's DE2 board

1 Getting Started

Each logic circuit, or subcircuit, being designed with Quartus II software is called a *project*. The software works on one project at a time and keeps all information for that project in a single directory (folder) in the file system. To begin a new logic circuit design, the first step is to create a directory to hold its files. To hold the design files for this tutorial, we will use a directory *Lab2*. The running example for this tutorial is a simple circuit for two-way light control.

Start the Quartus II 6 . 1 software. You should see a display similar to the one in Figure 2. This display consists of several windows that provide access to all the features of Quartus II software, which the user selects with the computer mouse. Most of the commands provided by Quartus II software can be accessed by using a set of menus that are located below the title bar. For example, in Figure 2 clicking the left mouse button on the menu named File opens the menu shown in Figure 3. Clicking the left mouse button on the entry Exit exits from Quartus II software. In general, whenever the mouse is used to select something, the *left* button is used. Hence we will not normally specify which button to press. In the few cases when it is necessary to use the *right* mouse button, it will be specified explicitly.

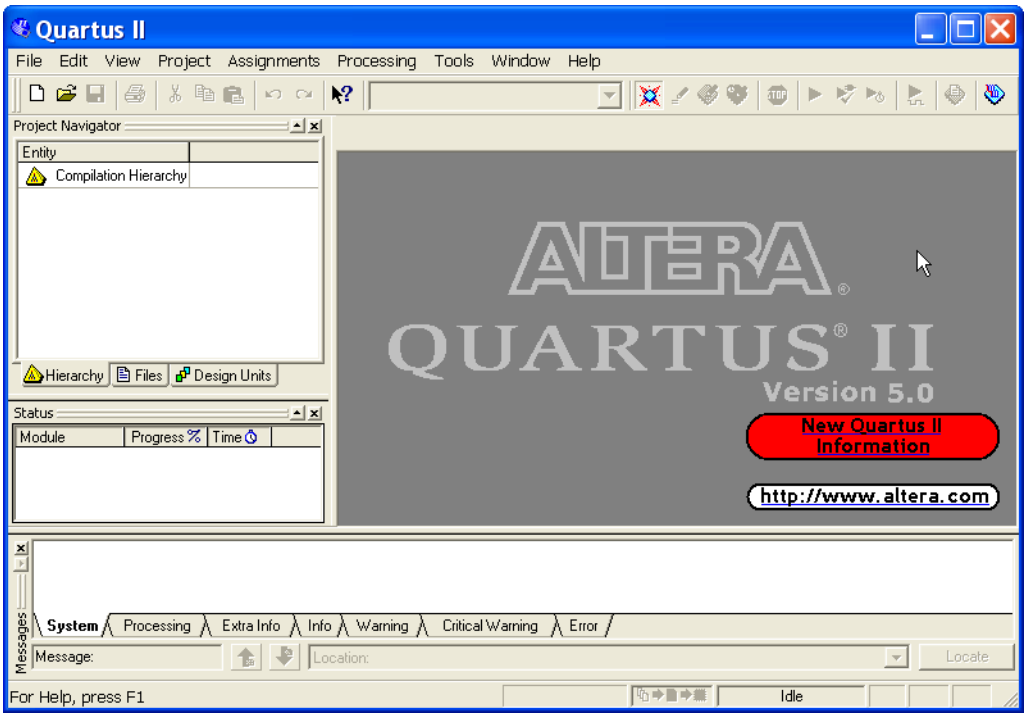


Figure 2. The main Quartus II display.

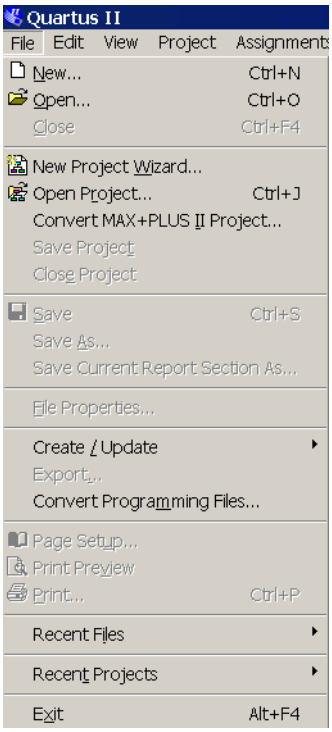


Figure 3. An example of the File menu.

For some commands it is necessary to access two or more menus in sequence. We use the convention Menu1 > Menu2 > Item to indicate that to select the desired command the user should first click the left mouse button on Menu1, then within this menu click on Menu2, and then within Menu2 click on Item. For example, File > Exit uses the mouse to exit from the system. Many commands can be invoked by clicking on an icon displayed in one of the toolbars. To see the command associated with an icon, position the mouse over the icon and a tooltip will appear that displays the command name.

1.1 Quartus II Online Help

Quartus II software provides comprehensive online documentation that answers many of the questions that may arise when using the software. The documentation is accessed from the menu in the Help window. To get some idea of the extent of documentation provided, it is worthwhile for the reader to browse through the Help menu. For instance, selecting Help > How to Use Help gives an indication of what type of help is provided.

The user can quickly search through the Help topics by selecting Help > Search, which opens a dialog box into which key words can be entered. Another method, context-sensitive help, is provided for quickly finding documentation for specific topics. While using most applications, pressing the F1 function key on the keyboard opens a Help display that shows the commands available for the application.

2 Starting a New Project

To start working on a new design we first have to define a new *design project*. Quartus II software makes the designer's task easy by providing support in the form of a *wizard*. Create a new project as follows:

1. Select File > New Project Wizard to reach the window in Figure 4, which indicates the capability of this wizard. You can skip this window in subsequent projects by checking the box Don't show me this introduction again. Press Next to get the window shown in Figure 5.

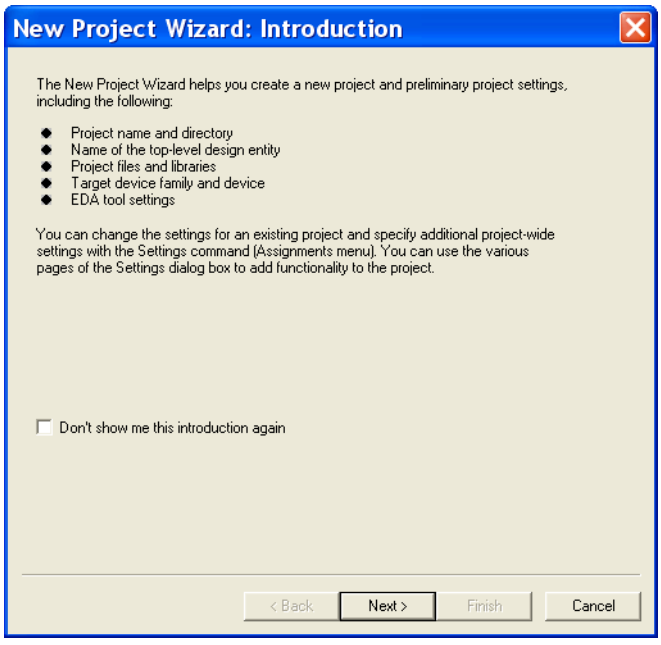


Figure 4. Tasks performed by the wizard.

Figure 5. Creation of a new project.

2. Set the working directory to be *Lab2*; of course, you can use some other directory name of your choice if you prefer. The project must have a name, which is usually the same as the top-level design entity that will be included in the project. Choose *light* as the name for both the project and the top-level entity, as shown in Figure 5. Press Next. Since we have not yet created the directory *Lab2*, Quartus II software displays the pop-up box in Figure 6 asking if it should create the desired directory. Click Yes, which leads to the window in Figure 7.

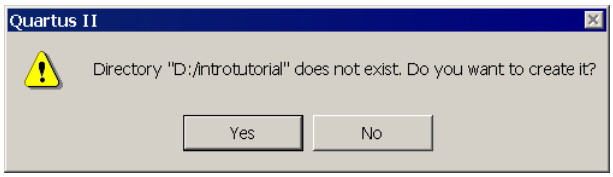


Figure 6. Quartus II software can create a new directory for the project.

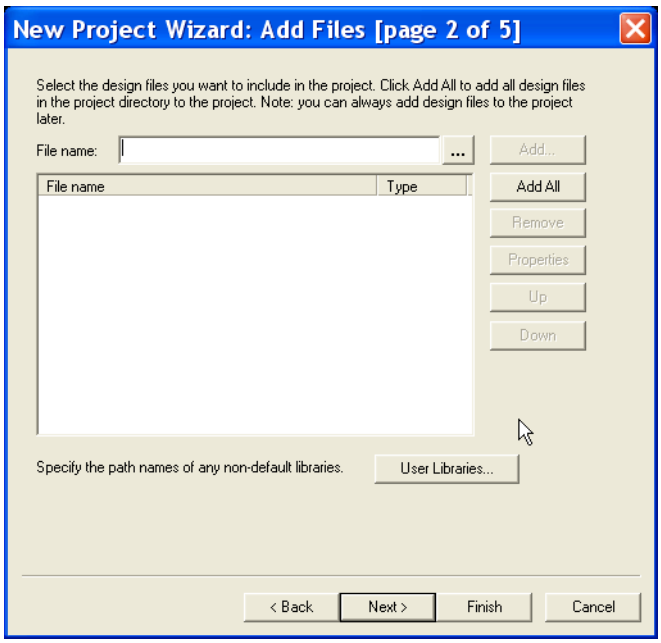


Figure 7. The wizard can include user-specified design files.

3. The wizard makes it easy to specify which existing files (if any) should be included in the project. Assuming that we do not have any existing files, click Next, which leads to the window in Figure 8.

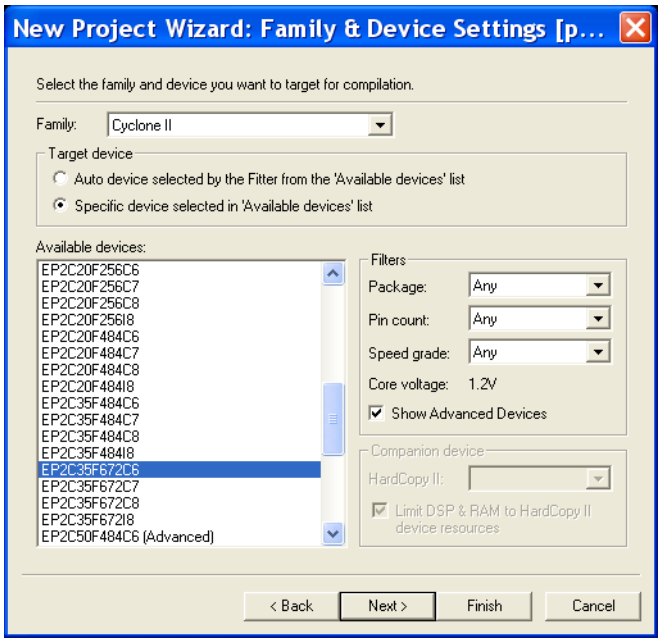


Figure 8. Choose the device family and a specific device.

4. We have to specify the type of device in which the designed circuit will be implemented. Choose Cyclone^{TM} II as the target device family. We can let Quartus II software select a specific device in the family, or we can choose the device explicitly. We will take the latter approach. From the list of available devices, choose the device called EP2C35F672C6 which is the FPGA used on Altera's DE2 board. Press Next, which opens the window in Figure 9.

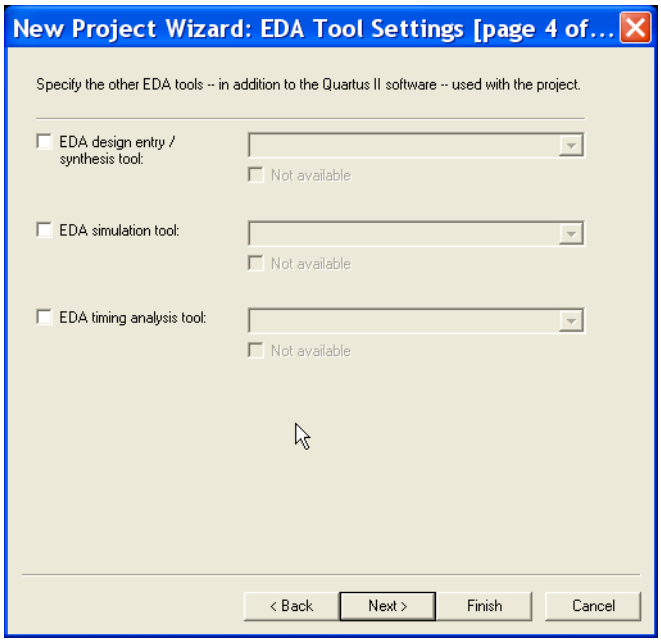


Figure 9. Other EDA tools can be specified.

5. The user can specify any third-party tools that should be used. A commonly used term for CAD software for electronic circuits is *EDA tools*, where the acronym stands for Electronic Design Automation. This term is used in Quartus II messages that refer to third-party tools, which are the tools developed and marketed by companies other than Altera. Since we will rely solely on Quartus II tools, we will not choose any other tools. Press Next.

6. A summary of the chosen settings appears in the screen shown in Figure 10. Press Finish, which returns to the main Quartus II window, but with *light* specified as the new project, in the display title bar, as indicated in Figure 11.

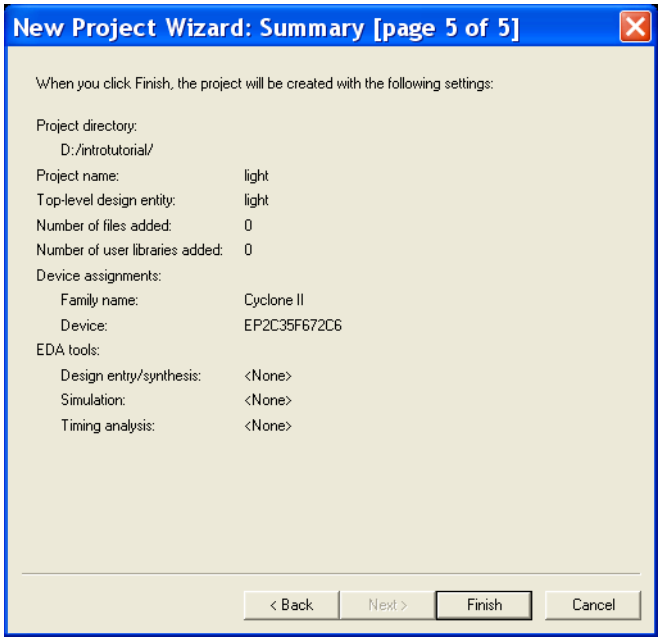


Figure 10. Summary of the project settings.

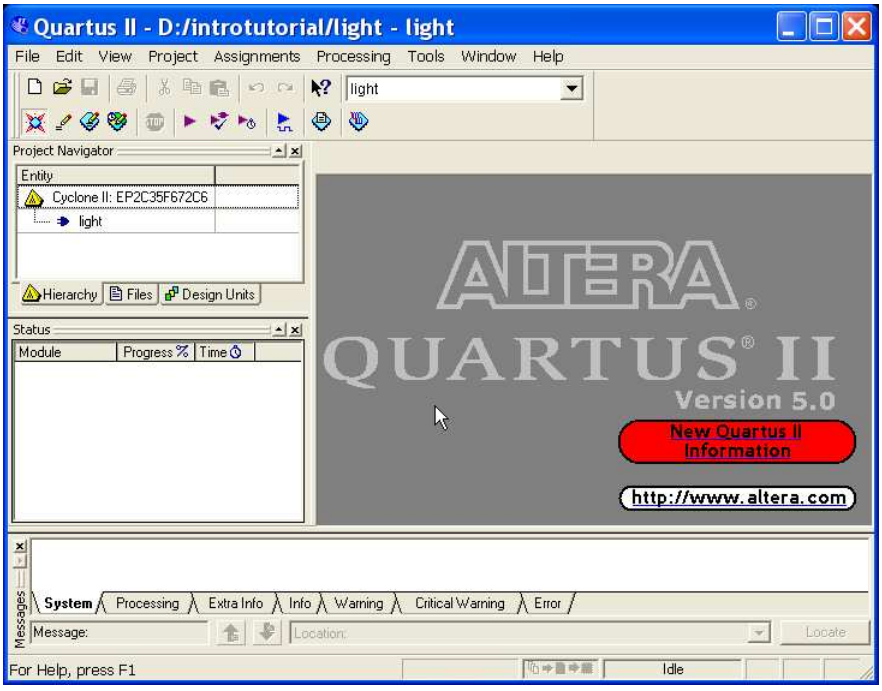
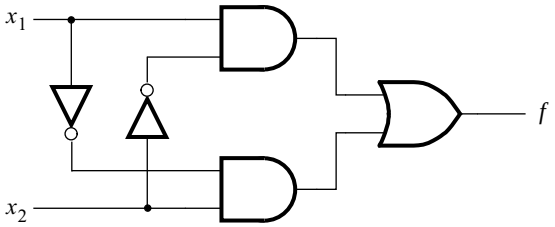


Figure 11. The Quartus II display for the created project.

3 Design Entry Using the Graphic Editor

As a design example, we will use the two-way light controller circuit shown in Figure 12. The circuit can be used to control a single light from either of the two switches, x_1 and x_2, where a closed switch corresponds to the logic value 1. The truth table for the circuit is also given in the figure. Note that this is just the Exclusive-OR function of the inputs x_1 and x_2, but we will implement it using the gates shown.



x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

Figure 12. The light controller circuit.

The Quartus II Graphic Editor can be used to specify a circuit in the form of a block diagram. Select File > New to get the window in Figure 13, choose Block Diagram/Schematic File, and click OK. This opens the Graphic Editor window. The first step is to specify a name for the file that will be created. Select File > Save As to open the pop-up box depicted in Figure 14. In the box labeled Save as type choose Block Diagram/Schematic File (*.bdf). In the box labeled File name type *light*, to match the name given in Figure 5, which was specified when the project was created. Put a checkmark in the box Add file to current project. Click Save, which puts the file into the directory L a b 1 and leads to the Graphic Editor window displayed in Figure 15.

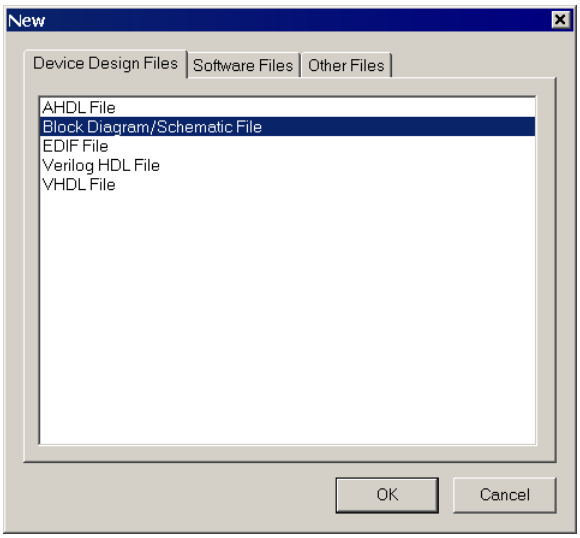


Figure 13. Choose to prepare a block diagram.

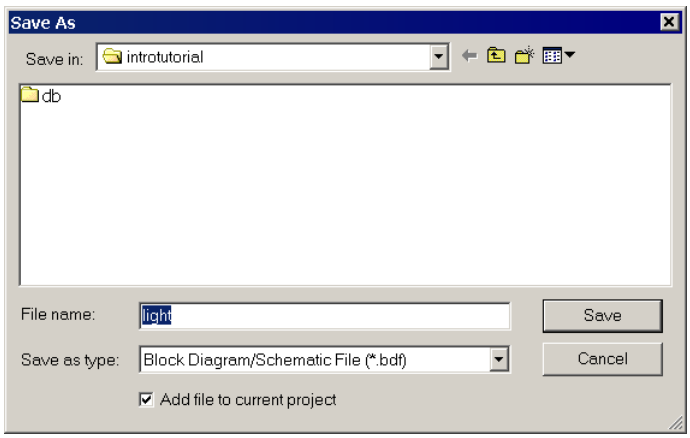


Figure 14. Name the file.

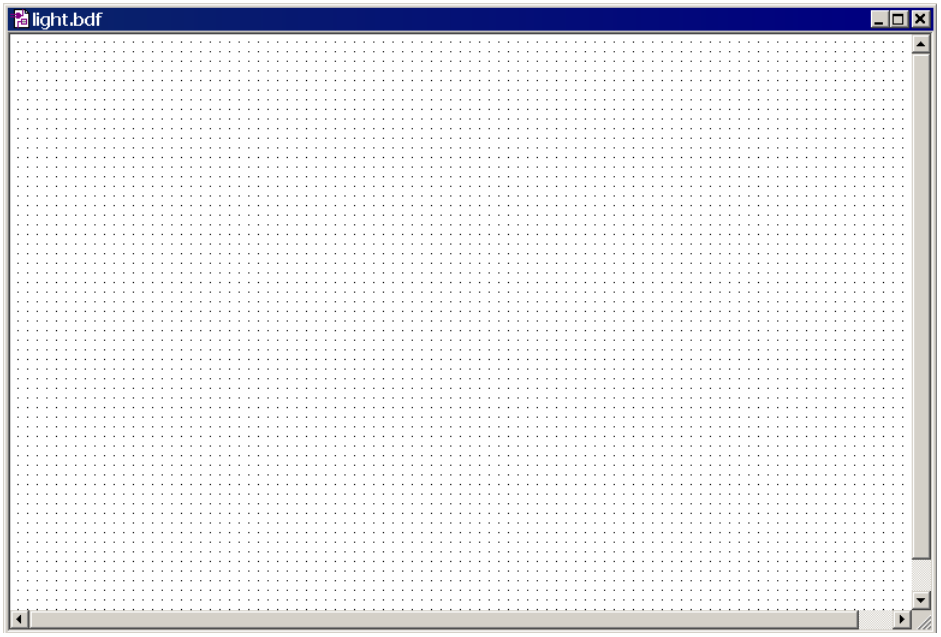


Figure 15. Graphic Editor window.

3.1 Importing Logic-Gate Symbols

The Graphic Editor provides a number of libraries which include circuit elements that can be imported into a schematic. Double-click on the blank space in the Graphic Editor window, or click on the icon in the toolbar that looks like an AND gate. A pop-up box in Figure 16 will appear. Expand the hierarchy in the Libraries box as shown in the figure. First expand *libraries*, then expand the library *primitives*, followed by expanding the library *logic* which comprises the logic gates. Select *and2*, which is a two-input AND gate, and click OK. Now, the AND gate symbol will appear in the Graphic Editor window. Using the mouse, move the symbol to a desirable location and click to place it there. Import the second AND gate, which can be done simply by positioning the mouse pointer over the existing AND-gate symbol, right-clicking, and dragging to make a copy of the symbol. A symbol in the Graphic Editor window can be moved by clicking on it and dragging it to a new location with the mouse

button pressed. Next, select *or2* from the library and import the OR gate into the diagram. Then, select *not* and import two instances of the NOT gate. Rotate the NOT gates into proper position by using the “Rotate left 90" icon  . Arrange the gates as shown in Figure 17.

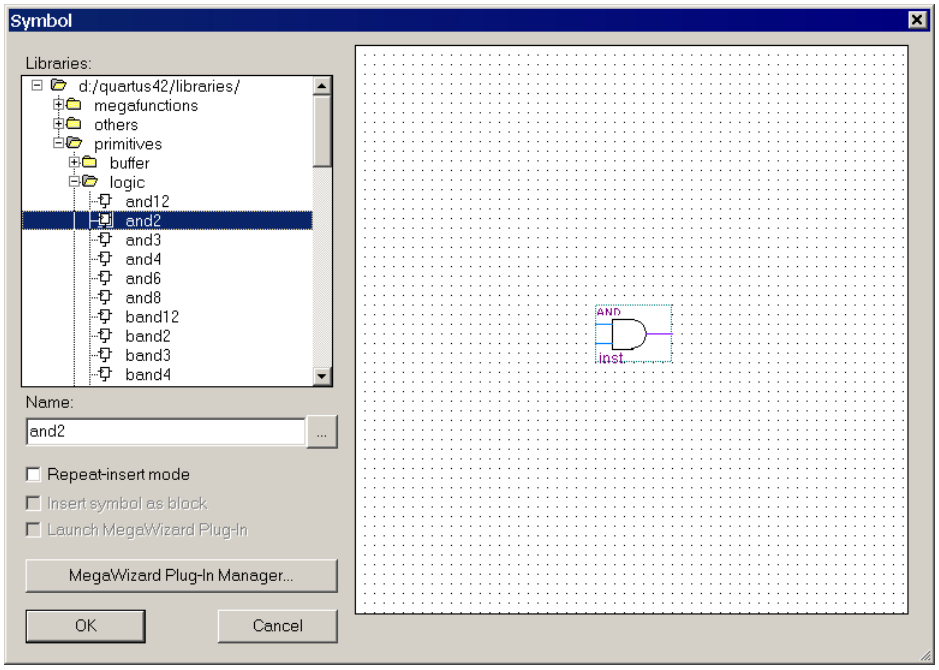


Figure 16. Choose a symbol from the library.

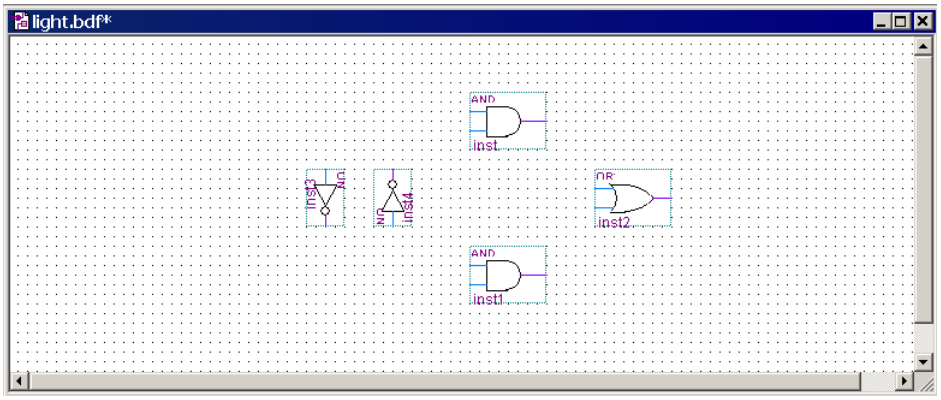


Figure 17. Import the gate symbols into the Graphic Editor window.

3.2 Importing Input and Output Symbols

Having entered the logic-gate symbols, it is now necessary to enter the symbols that represent the input and output ports of the circuit. Use the same procedure as for importing the gates, but choose the port symbols from the library *primitives/pin*. Import two instances of the input port and one instance of the output port, to obtain the image in Figure 18.

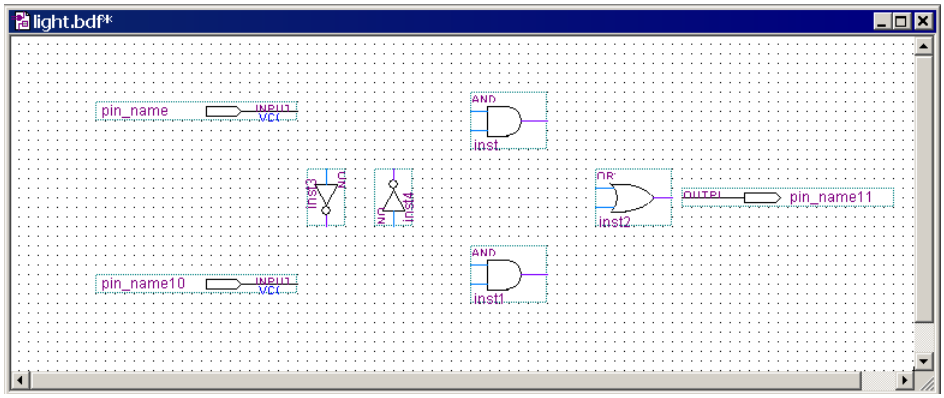


Figure 18. Import the input and output pins.

Assign names to the input and output symbols as follows. Point to the word *pin_name* on the top input symbol and double-click the mouse. The dialog box in Figure 19 will appear. Type the pin name, $x1$, and click OK. Similarly, assign the name $x2$ to the other input and f to the output.

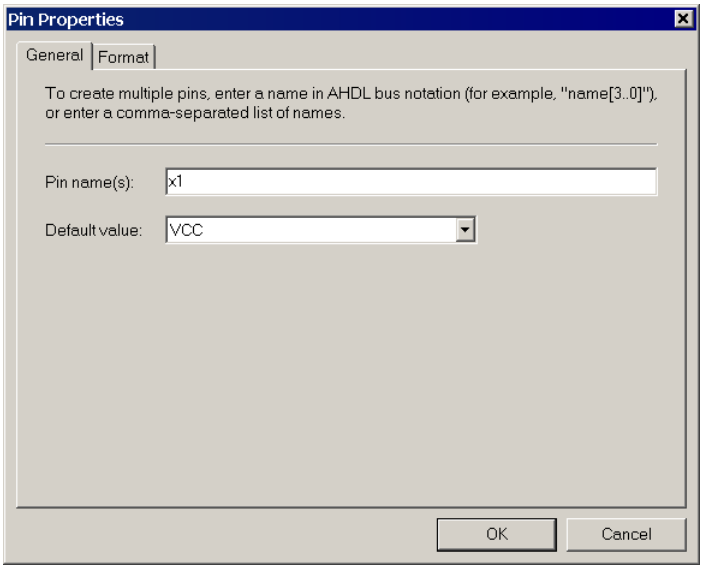


Figure 19. Naming of a pin.

3.3 Connecting Nodes with Wires

The symbols in the diagram have to be connected by drawing lines (wires). Click on the icon in the toolbar to activate the Orthogonal Node Tool. Position the mouse pointer over the right edge of the $x1$ input pin. Click and hold the mouse button and drag the mouse to the right until the drawn line reaches the pinstub on the top input of the AND gate. Release the mouse button, which leaves the line connecting the two pinstubs. Next, draw a wire from the input pinstub of the leftmost NOT gate to touch the wire that was drawn above it. Note that a dot will appear indicating a connection between the two wires.

Use the same procedure to draw the remaining wires in the circuit. If a mistake is made, a wire can be selected by clicking on it, and removed by pressing the Delete key on the keyboard. Upon completing the diagram, click

on the icon , to activate the Selection and Smart Drawing Tool. Now, changes in the appearance of the diagram can be made by selecting a particular symbol or wire and either moving it to a different location or deleting it. The final diagram is shown in Figure 20; save it.

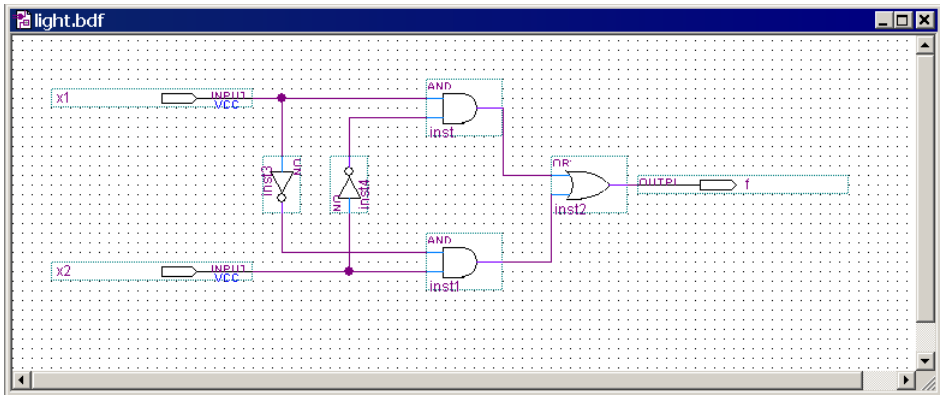


Figure 20. The completed schematic diagram.

4 Compiling the Designed Circuit

The entered schematic diagram file, *light.bdf*, is processed by several Quartus II tools that analyze the file, synthesize the circuit, and generate an implementation of it for the target chip. These tools are controlled by the application program called the *Compiler*.

Run the Compiler by selecting Processing > Start Compilation, or by clicking on the toolbar icon that looks like a purple triangle. As the compilation moves through various stages, its progress is reported in a window on the left side of the Quartus II display. Successful (or unsuccessful) compilation is indicated in a pop-up box. Acknowledge it by clicking OK, which leads to the Quartus II display in Figure 21. In the message window, at the bottom of the figure, various messages are displayed. In case of errors, there will be appropriate messages given.

When the compilation is finished, a compilation report is produced. A window showing this report is opened automatically, as seen in Figure 21. The window can be resized, maximized, or closed in the normal way, and it can be opened at any time either by selecting Processing > Compilation Report or by clicking on the icon . The report includes a number of sections listed on the left side of its window. Figure 21 displays the Compiler Flow Summary section, which indicates that only one logic element and three pins are needed to implement this tiny circuit on the selected FPGA chip. Another section is shown in Figure 22. It is reached by selecting Analysis & Synthesis > Equations on the left side of the compilation report. Here we see the logic expressions produced by the Compiler when synthesizing the designed circuit. Observe that f is the output derived as

$$f = x1 \$ x2$$

where the $ sign is used to represent the Exclusive-OR operation. Obviously, the Compiler recognized that the functionality of the circuit in our design file, *light.bdf*, can be represented by this expression.

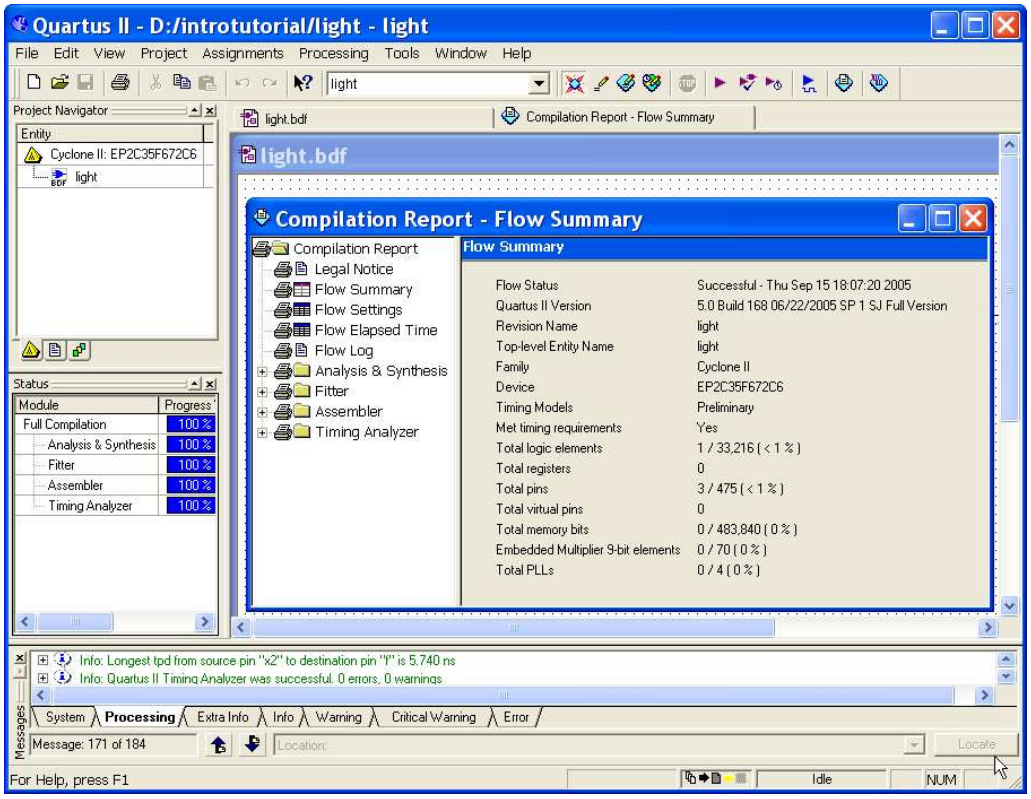


Figure 21. Display after a successful compilation.

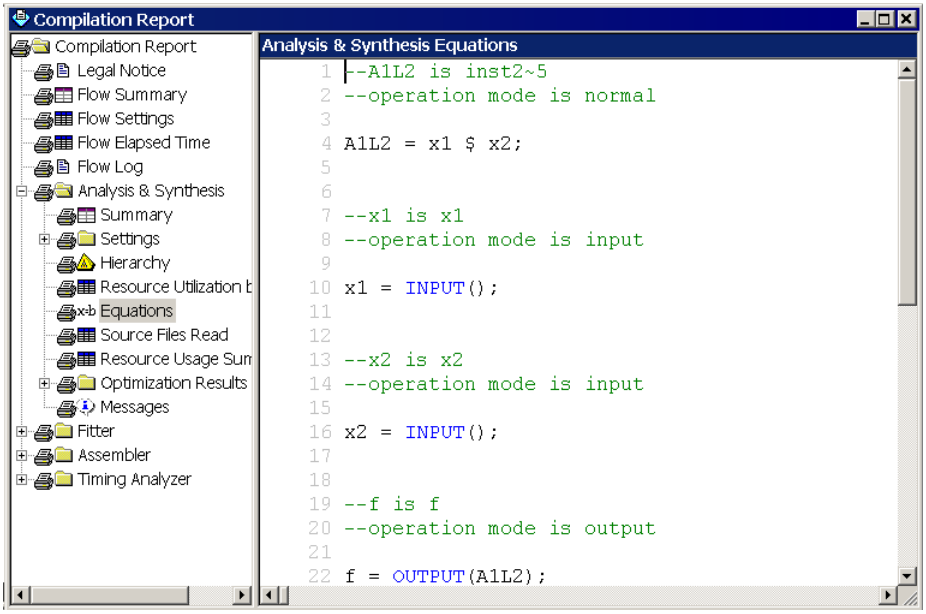


Figure 22. Compilation report showing the synthesized equations.

4.1 Errors

Quartus II software displays messages produced during compilation in the Messages window. If the block diagram design file is correct, one of the messages will state that the compilation was successful and that there are no errors.

If the Compiler does not report zero errors, then there is at least one mistake in the schematic entry. In this case a message corresponding to each error found will be displayed in the Messages window. Double-clicking on an error message will highlight the offending part of the circuit in the Graphic Editor window. Similarly, the Compiler may display some warning messages. Their details can be explored in the same way as in the case of error messages. The user can obtain more information about a specific error or warning message by selecting the message and pressing the F1 function key.

To see the effect of an error, open the file *light.bdf*. Remove the wire connecting the output of the top AND gate to the OR gate. To do this, click on the icon, click the mouse on the wire to be removed (to select it) and press Delete. Compile the erroneous design by clicking on the icon. A pop-up box will ask if the changes made to the *light.bdf* file should be saved; click Yes. After trying to compile the circuit, Quartus II software will display a pop-up box indicating that the compilation was not successful. Acknowledge it by clicking OK. The compilation report summary, given in Figure 23, now confirms the failed result. Expand the Analysis & Synthesis part of the report and then select Messages to have the messages displayed as shown in Figure 24. Double-click on the first error message, which states that one of the nodes is missing a source. Quartus II software responds by displaying the *light.bdf* schematic and highlighting the OR gate which is affected by the error, as shown in Figure 25. Correct the error and recompile the design.

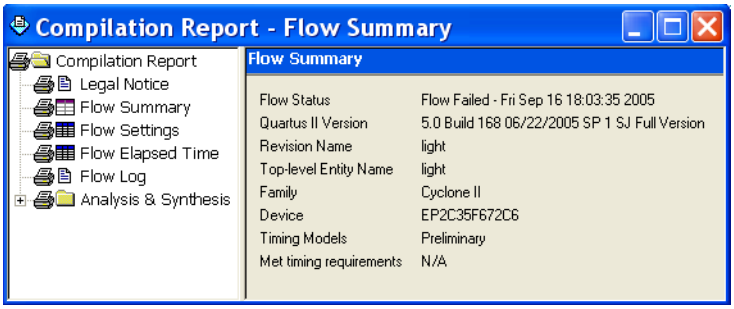


Figure 23. Compilation report for the failed design.

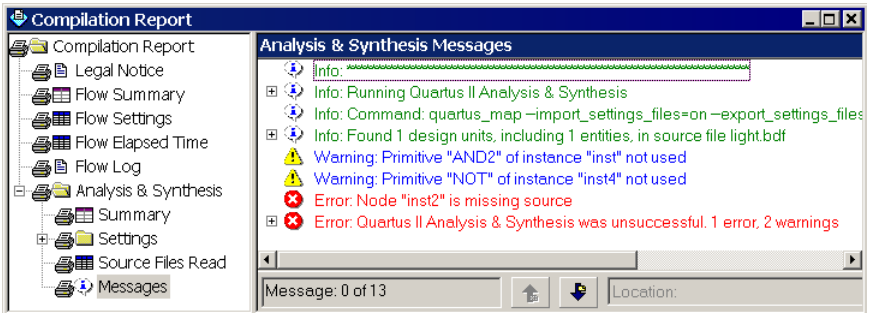


Figure 24. Error messages.

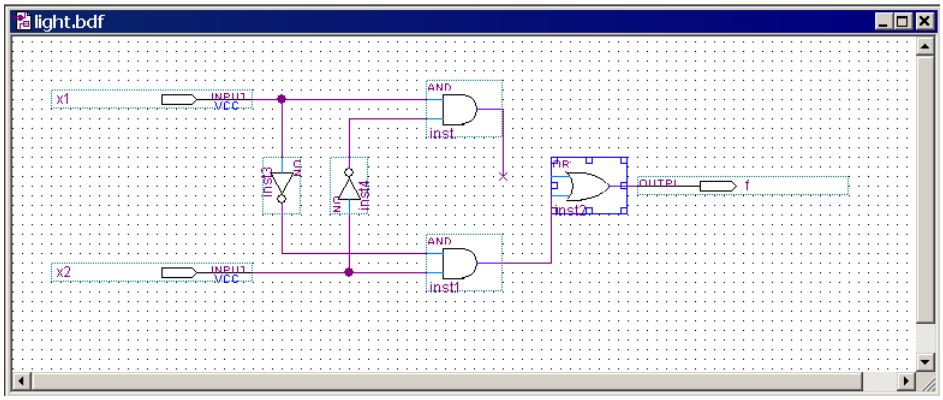


Figure 25. Identifying the location of the error.

5 Pin Assignment

During the compilation above, the Quartus II Compiler was free to choose any pins on the selected FPGA to serve as inputs and outputs. However, the DE2 board has hardwired connections between the FPGA pins and the other components on the board. We will use two toggle switches, labeled SW_0 and SW_1, to provide the external inputs, x_1 and x_2, to our example circuit. These switches are connected to the FPGA pins N25 and N26, respectively. We will connect the output to the green light-emitting diode labeled $LEDG_0$, which is hardwired to the FPGA pin AE22. Pin assignments for the switches, LEDs & 7-Segment display are attached at the end of the document.

Figure 26. The Assignment Editor window.

Pin assignments are made by using the Assignment Editor. Select Assignments > Pins to reach the window in Figure 26. Under Category select Pin. ~~Double-click on the entry <<new>> which is highlighted in blue in the column labeled To. The drop-down menu in Figure 27 will appear.~~ Click on $x1$ as the first pin to be assigned; this will enter $x1$ in the displayed table. Follow this by double-clicking on the box to the right of this new $x1$

entry, in the column labeled Location. Now, the drop-down menu in Figure 28 appears. Scroll down and select PIN_N25. Instead of scrolling down the menu to find the desired pin, you can just type the name of the pin (N25) in the Location box. Use the same procedure to assign input $x2$ to pin N26 and output f to pin AE22, which results in the image in Figure 29. To save the assignments made, choose File > Save. You can also simply close the Assignment Editor window, in which case a pop-up box will ask if you want to save the changes to assignments; click Yes. Recompile the circuit, so that it will be compiled with the correct pin assignments.

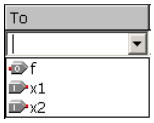


Figure 27. The drop-down menu displays the input and output names.

Location	I/O Bank	I/O Standard	General Function	Special Fun
		LVTTL		
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	LVDS31p	
PIN_N18	I/O Bank 5	Row I/O	LVDS110p	
PIN_N20	I/O Bank 5	Row I/O	LVDS124p	
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	LVDS126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	
PIN_P1	I/O Bank 1	Dedicated Clock	CLK3, LVDSCLK1n, Input	
PIN_P2	I/O Bank 1	Dedicated Clock	CLK2, LVDSCLK1p, Input	
PIN_P3	I/O Bank 1	Row I/O	LVDS26p, DPCLK1/DQ51L/CQ1L#	
PIN_P4	I/O Bank 1	Row I/O	LVDS26n	
PIN_P6	I/O Bank 1	Row I/O	LVDS22n	
PIN_P7	I/O Bank 1	Row I/O	LVDS22p	
PIN_P9	I/O Bank 2	Row I/O	LVDS31n	
PIN_P17	I/O Bank 6	Row I/O	LVDS130n	
PIN_P18	I/O Bank 5	Row I/O	LVDS110n	
PIN_P23	I/O Bank 6	Row I/O	LVDS127p, DPCLK6/DQS1R/CQ1R#	
PIN_P24	I/O Bank 6	Row I/O	LVDS127n	

Figure 28. The available pins.

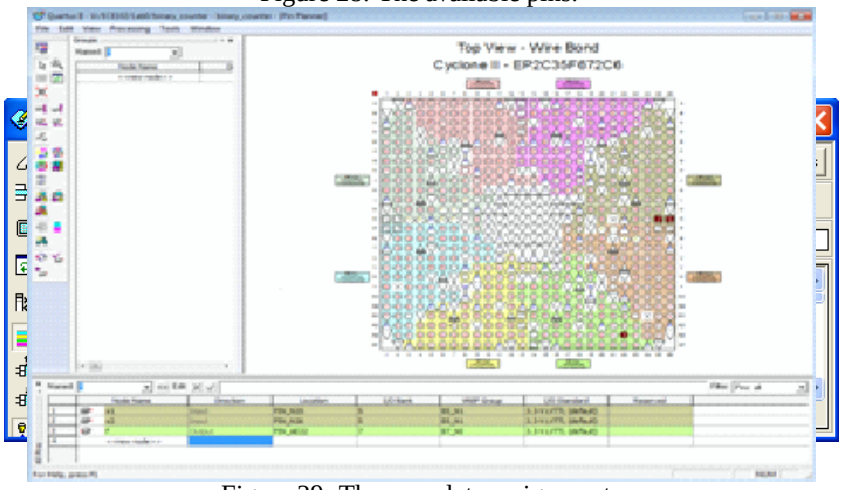


Figure 29. The complete assignment.

The DE2 board has fixed pin assignments. Having finished one design, the user will want to use the same pin assignment for subsequent designs. Going through the procedure described above becomes tedious if there are many pins used in the design. A useful Quartus II feature allows the user to both export and import the pin assignments from a special file format, rather than creating them manually using the Assignment Editor. A simple

file format that can be used for this purpose is the *comma separated value (CSV)* format, which is a common text file format that contains comma-delimited values. This file format is often used in conjunction with the Microsoft Excel spreadsheet program, but the file can also be created by hand using any plain ASCII text editor. The format for the file for our simple project is

```
To, Location
x1, PIN_N25
x2, PIN_N26
f, PIN_AE22
```

By adding lines to the file, any number of pin assignments can be created. Such *csv* files can be imported into any design project.

If you created a pin assignment for a particular project, you can export it for use in a different project. To see how this is done, open again the Assignment Editor to reach the window in Figure 29. Now, select File > Export which leads to the window in Figure 30. Here, the file *light.csv* is available for export. Click on Export. If you now look in the directory *Lab1*, you will see that the file *light.csv* has been created.

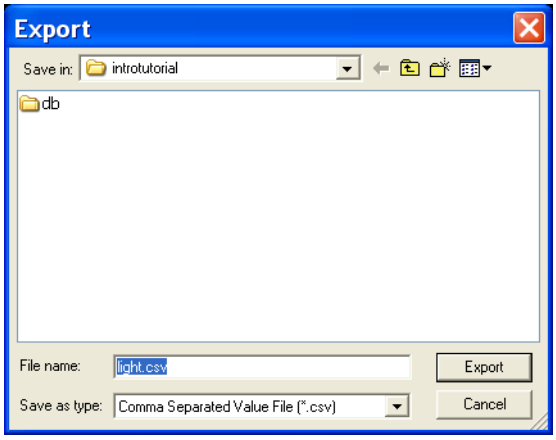


Figure 30. Exporting the pin assignment.

You can import a pin assignment by choosing Assignments > Import Assignments. This opens the dialogue in Figure 31 to select the file to import. Type the name of the file, including the *csv* extension and the full path to the directory that holds the file, in the File Name box and press OK. Of course, you can also browse to find the desired file.

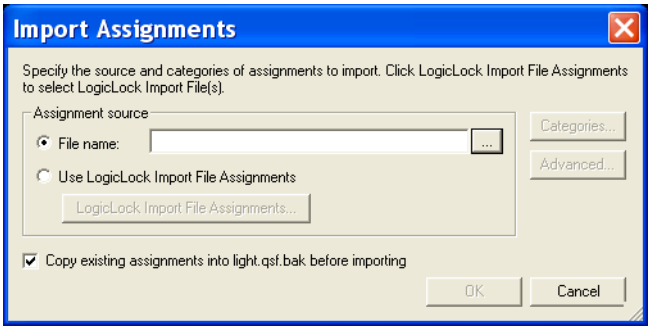


Figure 31. Importing the pin assignment.

For convenience when using large designs, all relevant pin assignments for the DE2 board are given in the file called *DE2_pin_assignments.csv* in the directory *DE2_tutorials\design_files*, which is included on the CD-ROM that accompanies the DE2 board and can also be found on Altera's DE2 web pages. This file uses the names found in the *DE2 User Manual*. If we wanted to make the pin assignments for our example circuit by importing this file, then we would have to use the same names in our Block Diagram/Schematic design file; namely, *SW[0]*, *SW[1]* and *LEDG[0]* for *x1*, *x2* and *f*, respectively. Since these signals are specified in the *DE2_pin_assignments.csv* file as elements of vectors *SW* and *LEDG*, we must refer to them in the same way in our design file. For example, in the *DE2_pin_assignments.csv* file the 18 toggle switches are called *SW[17]* to *SW[0]*. In a design file they can also be referred to as a vector *SW[17..0]*.

6 Simulating the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE2 board, it is prudent to simulate it to ascertain its correctness. Quartus II software includes a simulation tool that can be used to simulate the behavior of a designed circuit. Before the circuit can be simulated, it is necessary to create the desired waveforms, called *test vectors*, to represent the input signals. It is also necessary to specify which outputs, as well as possible internal points in the circuit, the designer wishes to observe. The simulator applies the test vectors to a model of the implemented circuit and determines the expected response. We will use the Quartus II Waveform Editor to draw the test vectors, as follows:

1. Open the Waveform Editor window by selecting File > New, which gives the window shown in Figure 32. Click on the Other Files tab to reach the window displayed in Figure 33. Choose Vector Waveform File and click OK.

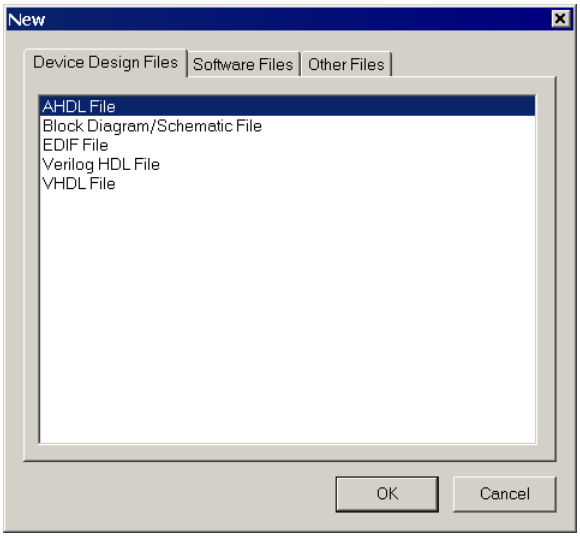


Figure 32. Need to prepare a new file.

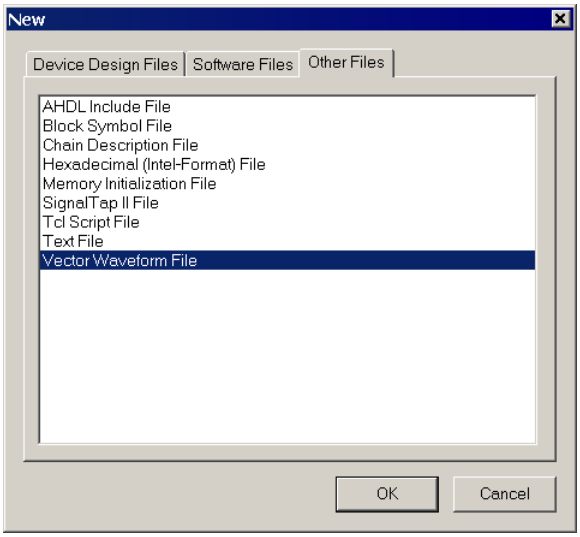


Figure 33. Choose to prepare a test-vector file.

2. The Waveform Editor window is depicted in Figure 34. Save the file under the name *light.vwf*; note that this changes the name in the displayed window. Set the desired simulation to run from 0 to 200 ns by selecting Edit > End Time and entering 200 ns in the dialog box that pops up. Selecting View > Fit in Window displays the entire simulation range of 0 to 200 ns in the window, as shown in Figure 35. You may wish to resize the window to its maximum size.

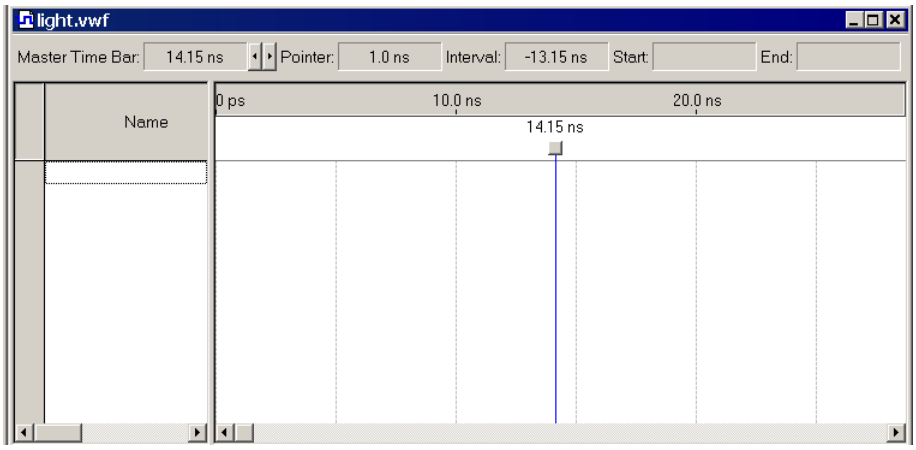


Figure 34. The Waveform Editor window.

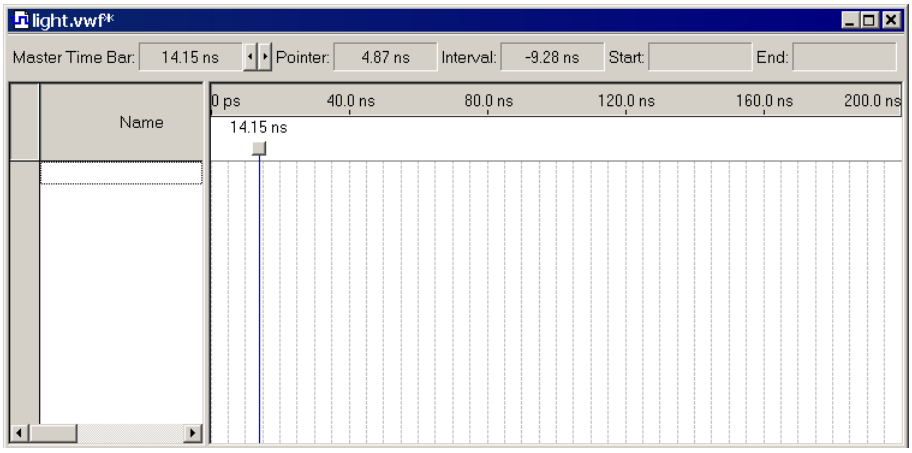


Figure 35. The augmented Waveform Editor window.

3. Next, we want to include the input and output nodes of the circuit to be simulated. Click Edit > Insert Node or Bus to open the window in Figure 36. It is possible to type the name of a signal (pin) into the Name box, but it is easier to click on the button labeled Node Finder to open the window in Figure 37. The Node Finder utility has a filter used to indicate what type of nodes are to be found. Since we are interested in input and output pins, set the filter to Pins: all. Click the List button to find the input and output nodes as indicated on the left side of the figure.

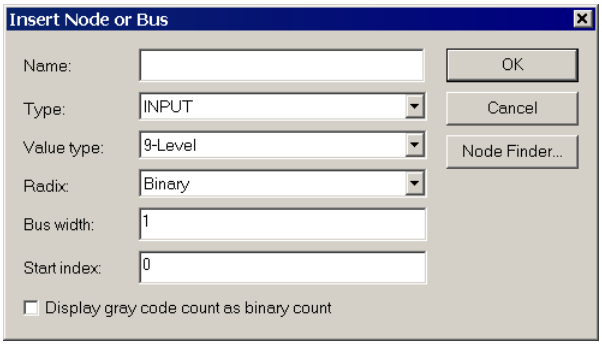


Figure 36. The Insert Node or Bus dialogue.

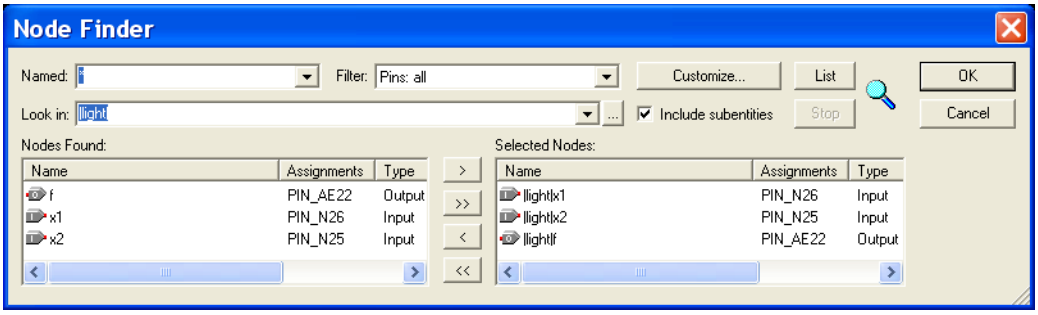


Figure 37. Selecting nodes to insert into the Waveform Editor.

Click on the *x1* signal in the Nodes Found box in Figure 37, and then click the > sign to add it to the Selected Nodes box on the right side of the figure. Do the same for *x2* and *f*. Click OK to close the Node Finder window, and then click OK in the window of Figure 36. This leaves a fully displayed Waveform Editor window, as shown in Figure 38. If you did not select the nodes in the same order as displayed in Figure 38, it is possible to rearrange them. To move a waveform up or down in the Waveform Editor window, click on the node name (in the Name column) and release the mouse button. The waveform is now highlighted to show the selection. Click again on the waveform and drag it up or down in the Waveform Editor.

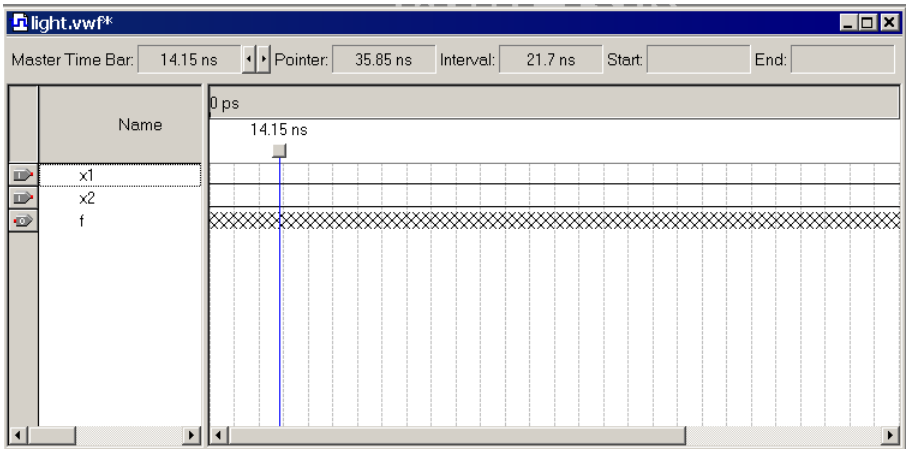


Figure 38. The nodes needed for simulation.

4. We will now specify the logic values to be used for the input signals *x1* and *x2* during simulation. The logic values at the output *f* will be generated automatically by the simulator. To make it easy to draw the desired waveforms, the Waveform Editor displays (by default) vertical guidelines and provides a drawing feature that snaps on these lines (which can otherwise be invoked by choosing View > Snap to Grid). Observe also a solid vertical line, which can be moved by pointing to its top and dragging it horizontally. This reference line is used in analyzing the timing of a circuit; move it to the *time* = 0 position. The waveforms can be drawn using the Selection Tool, which is activated by selecting the icon in the toolbar, or the Waveform Editing Tool, which is activated by the icon .

   To simulate the behavior of a large circuit, it is necessary to apply a sufficient number of input valuations and observe the expected values of the outputs. In a large circuit the number of possible input valuations may be huge, so in practice we choose a relatively small (but representative) sample of these input valuations. However, for our tiny circuit we can simulate all four input valuations given in Figure 12. We will use four 50-ns time intervals to apply the four test vectors.

   We can generate the desired input waveforms as follows. Click on the waveform name for the *x1* node. Once a waveform is selected, the editing commands in the Waveform Editor can be used to draw the desired waveforms. Commands are available for setting a selected signal to 0, 1, unknown (X), high impedance (Z), don't care (DC), inverting its existing value (INV), or defining a clock waveform. Each command can be activated by using the Edit > Value command, or via the toolbar for the Waveform Editor. The Edit menu can also be opened by right-clicking on a waveform name.

   Set *x1* to 0 in the time interval 0 to 100 ns, which is probably already set by default. Next, set *x1* to 1 in the time interval 100 to 200 ns. Do this by pressing the mouse at the start of the interval and dragging it to its end, which highlights the selected interval, and choosing the logic value 1 in the toolbar. Make *x2* = 1 from 50 to 100 ns and also from 150 to 200 ns, which corresponds to the truth table in Figure 12. This should

produce the image in Figure 39. Observe that the output f is displayed as having an unknown value at this time, which is indicated by a hashed pattern; its value will be determined during simulation. Save the file.

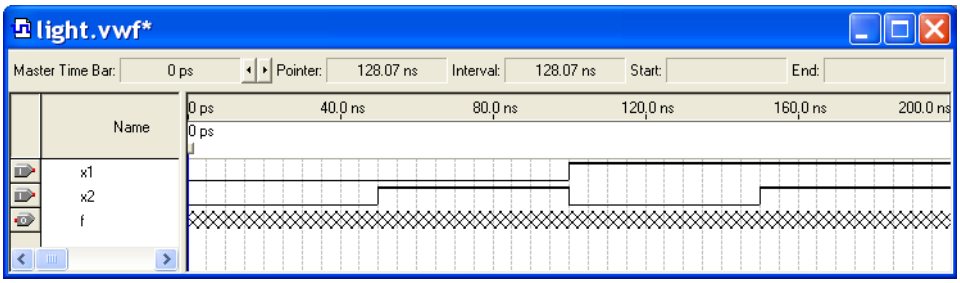


Figure 39. Setting of test values.

6.1 Performing the Simulation

A designed circuit can be simulated in two ways. The simplest way is to assume that logic elements and interconnection wires in the FPGA are perfect, thus causing no delay in propagation of signals through the circuit. This is called *functional simulation*. A more complex alternative is to take all propagation delays into account, which leads to *timing simulation*. Typically, functional simulation is used to verify the functional correctness of a circuit as it is being designed. This takes much less time, because the simulation can be performed simply by using the logic expressions that define the circuit.

6.1.1 Functional Simulation

To perform the functional simulation, select Assignments > Settings to open the Settings window. On the left side of this window click on Simulator to display the window in Figure 40, choose Functional as the simulation mode, and click OK. The Quartus II simulator takes the inputs and generates the outputs defined in the *light.vwf* file. Before running the functional simulation it is necessary to create the required netlist, which is done by selecting Processing > Generate Functional Simulation Netlist. A simulation run is started by Processing > Start Simulation, or by using the icon . At the end of the simulation, Quartus II software indicates its successful completion and displays a Simulation Report illustrated in Figure 41. If your report window does not show the entire simulation time range, click on the report window to select it and choose View > Fit in Window. Observe that the output f is as specified in the truth table of Figure 12.

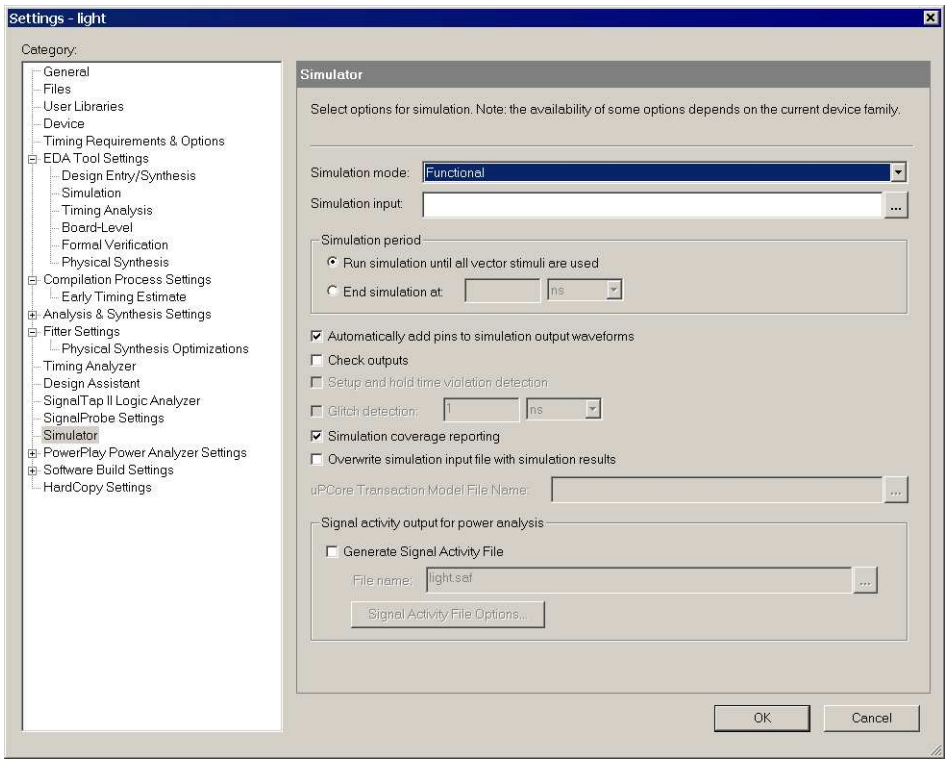


Figure 40. Specifying the simulation mode.

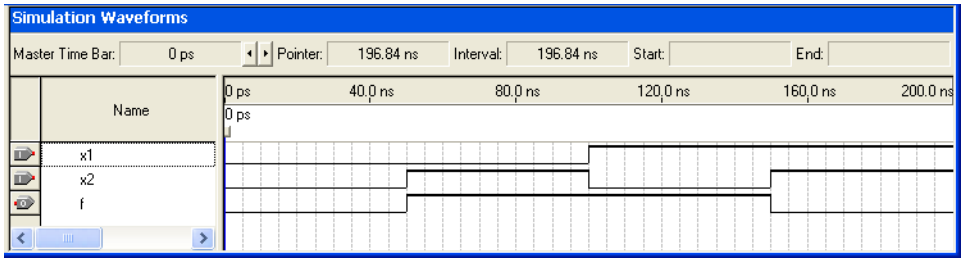


Figure 41. The result of functional simulation.

6.1.2 Timing Simulation

Having ascertained that the designed circuit is functionally correct, we should now perform the timing simulation to see how it will behave when it is actually implemented in the chosen FPGA device. Select Assignments > Settings > Simulator to get to the window in Figure 40, choose Timing as the simulation mode, and click OK. Run the simulator, which should produce the waveforms in Figure 42. Observe that there is a delay of about 6 ns in producing a change in the signal f from the time when the input signals, x_1 and x_2, change their values. This delay is due to the propagation delays in the logic element and the wires in the FPGA device. You may also notice that a momentary change in the value of f, from 1 to 0 and back to 1, occurs at about 106-ns point in the simulation. This *glitch* is also due to the propagation delays in the FPGA device, because changes in x_1 and x_2 may not arrive at exactly the same time at the logic element that generates f.

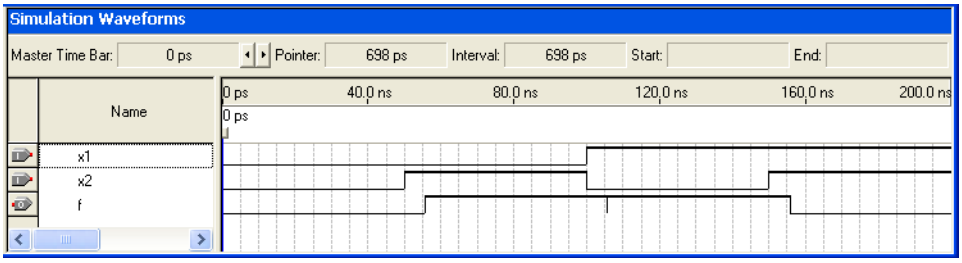


Figure 42. The result of timing simulation.

7 Programming and Configuring the FPGA Device

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus II Compiler's Assembler module. Altera's DE2 board allows the configuration to be done in two different ways, known as JTAG and AS modes. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board. To use this connection, it is necessary to have the USB-Blaster driver installed. If this driver is not already installed, consult the tutorial *Getting Started with Altera's DE2 Board* for information about installing the driver. Before using the board, make sure that the USB cable is properly connected and turn on the power supply switch on the board.

In the JTAG mode, the configuration data is loaded directly into the FPGA device. The acronym JTAG stands for Joint Test Action Group. This group defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. If the FPGA is configured in this manner, it will retain its configuration as long as the power remains turned on. The configuration information is lost when the power is turned off. The second possibility is to use the Active Serial (AS) mode. In this case, a configuration device that includes some flash memory is used to store the configuration data. Quartus II software places the configuration data into the configuration device on the DE2 board. Then, this data is loaded into the FPGA upon power-up or reconfiguration. Thus, the FPGA need not be configured by the Quartus II software if the power is turned off and on. The choice between the two modes is made by the RUN/PROG switch on the DE2 board. The RUN position selects the JTAG mode, while the PROG position selects the AS mode.

7.1 JTAG Programming

The programming and configuration task is performed as follows. Flip the RUN/PROG switch into the RUN position. Select Tools > Programmer to reach the window in Figure 43. Here it is necessary to specify the programming hardware and the mode that should be used. If not already chosen by default, select JTAG in the Mode box. Also, if the USB-Blaster is not chosen by default, press the Hardware Setup... button and select the USB-Blaster in the window that pops up, as shown in Figure 44.

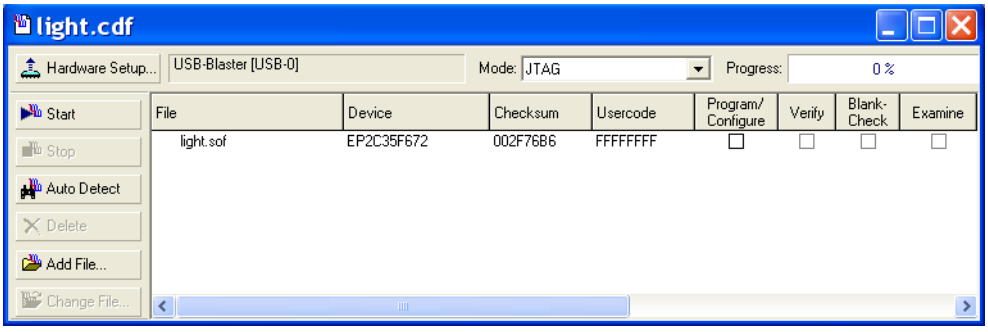


Figure 43. The Programmer window.

Observe that the configuration file *light.sof* is listed in the window in Figure 43. If the file is not already listed, then click Add File and select it. This is a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device. The extension *.sof* stands for SRAM Object File. Note also that the device selected is EP2C35F672, which is the FPGA device used on the DE2 board. Click on the Program/Configure check box, as shown in Figure 45.

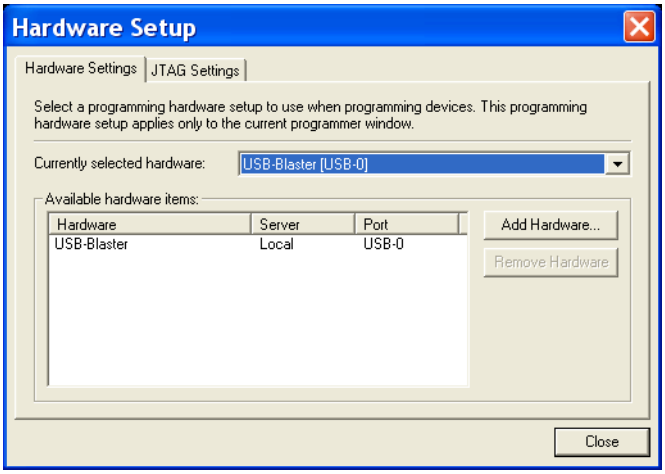


Figure 44. The Hardware Setup window.

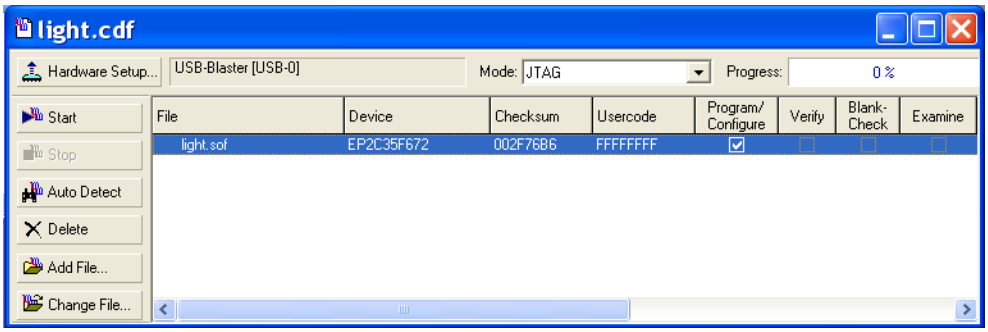


Figure 45. The updated Programmer window.

Now, press Start in the window in Figure 45. An LED on the board will light up when the configuration data has been downloaded successfully. If you see an error reported by Quartus II software indicating that programming failed, then check to ensure that the board is properly powered on.

7.2 Active Serial Mode Programming

In this case, the configuration data has to be loaded into the configuration device on the DE2 board, which is identified by the name EPCS16. To specify the required configuration device select Assignments > Device, which leads to the window in Figure 46. Click on the Device & Pin Options button to reach the window in Figure 47. Now, click on the Configuration tab to obtain the window in Figure 48. In the Configuration device box (which may be set to Auto) choose EPCS16 and click OK. Upon returning to the window in Figure 46, click OK. Recompile the designed circuit.

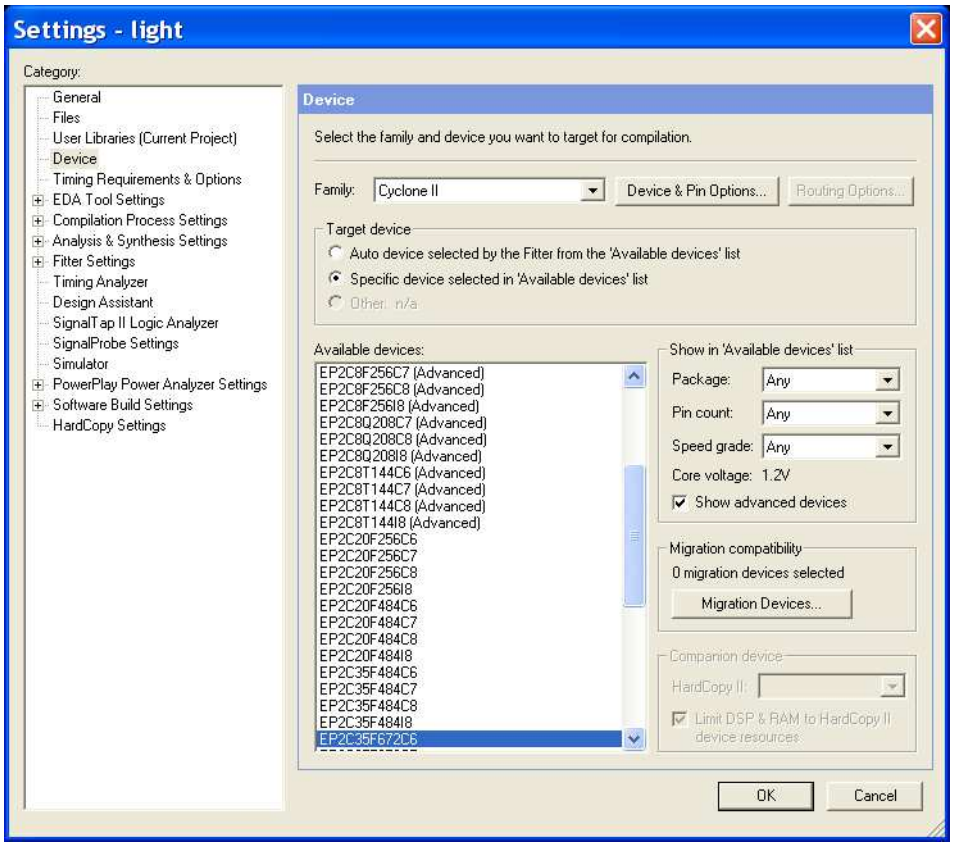


Figure 46. The Device Settings window.

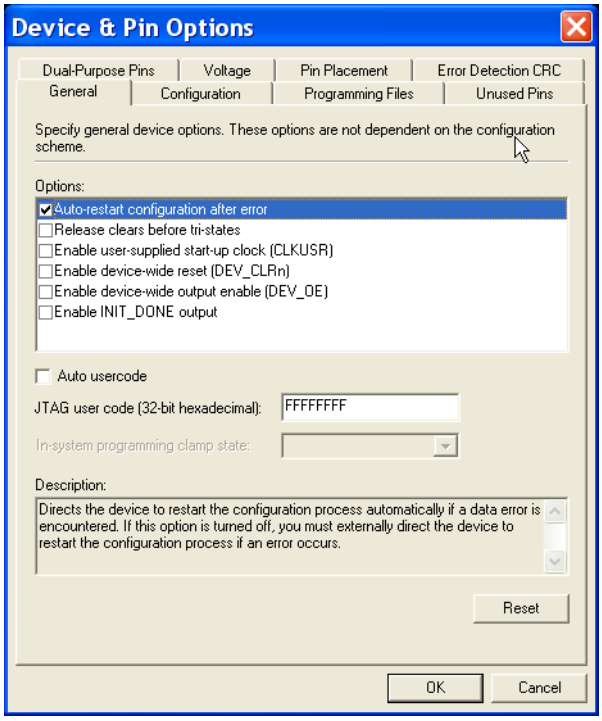


Figure 47. The Options window.

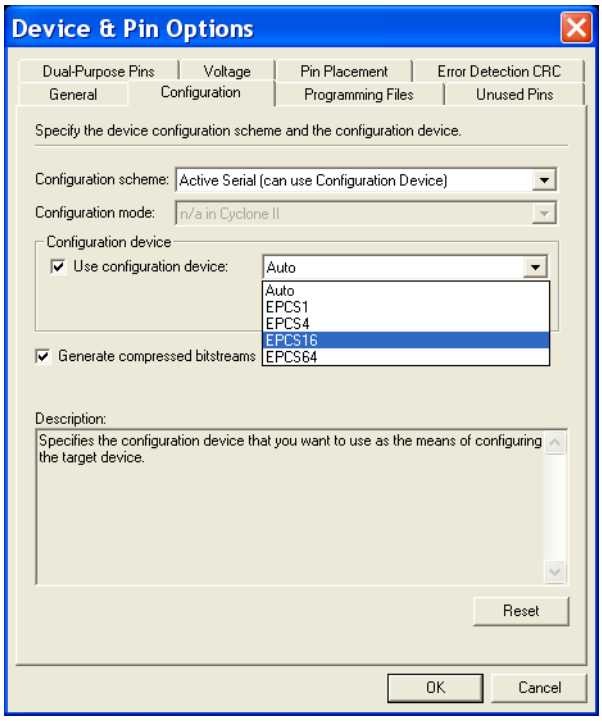


Figure 48. Specifying the configuration device.

The rest of the procedure is similar to the one described above for the JTAG mode. Select Tools > Programmer to reach the window in Figure 43. In the Mode box select Active Serial Programming. If you are changing the mode from the previously used JTAG mode, the pop-up box in Figure 49 will appear, asking if you want to clear all devices. Click Yes. Now, the Programmer window shown in Figure 50 will appear. Make sure that the Hardware Setup indicates the USB-Blaster. If the configuration file is not already listed in the window, press Add File. The pop-up box in Figure 51 will appear. Select the file *light.pof* in the directory *Lab1* and click Open. As a result, the configuration file *light.pof* will be listed in the window. This is a binary file produced by the Compiler's Assembler module, which contains the data to be loaded into the EPCS16 configuration device. The extension *.pof* stands for Programmer Object File. Upon returning to the Programmer window, click on the Program/Configure check box, as shown in Figure 52.

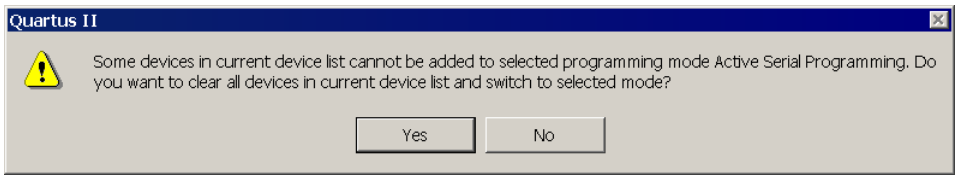


Figure 49. Clear the previously selected devices.

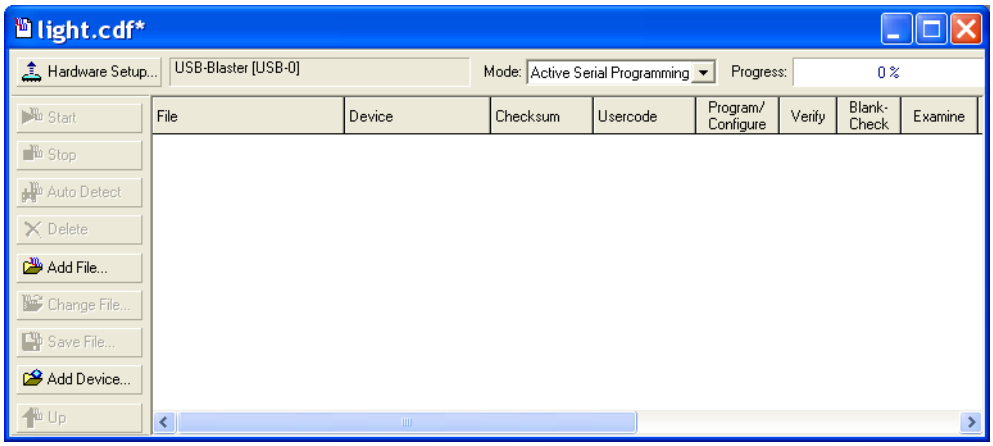


Figure 50. The Programmer window with Active Serial Programming selected.

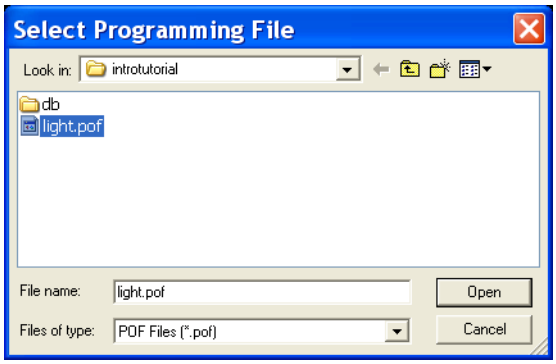


Figure 51. Choose the configuration file.

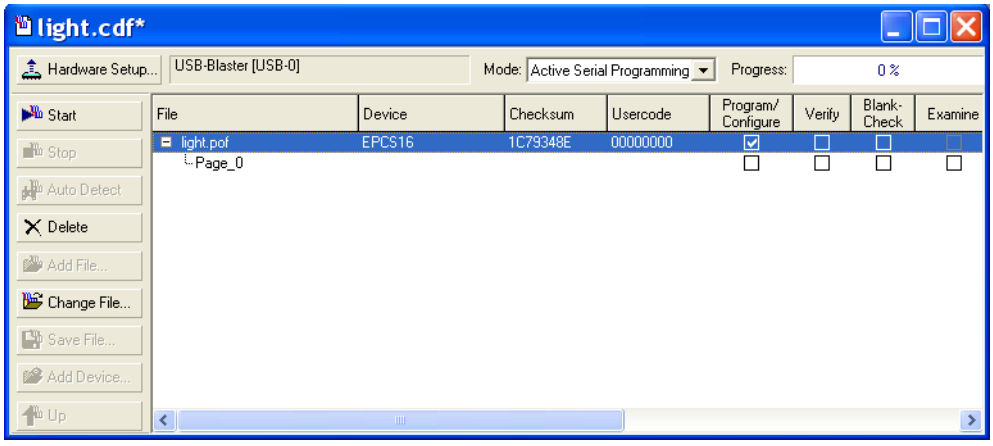


Figure 52. The updated Programmer window.

Flip the RUN/PROG switch on the DE2 board to the PROG position. Press Start in the window in Figure 52. An LED on the board will light up when the configuration data has been downloaded successfully. Also, the

Progress box in Figure 52 will indicate when the configuration and programming process is completed, as shown in Figure 53.

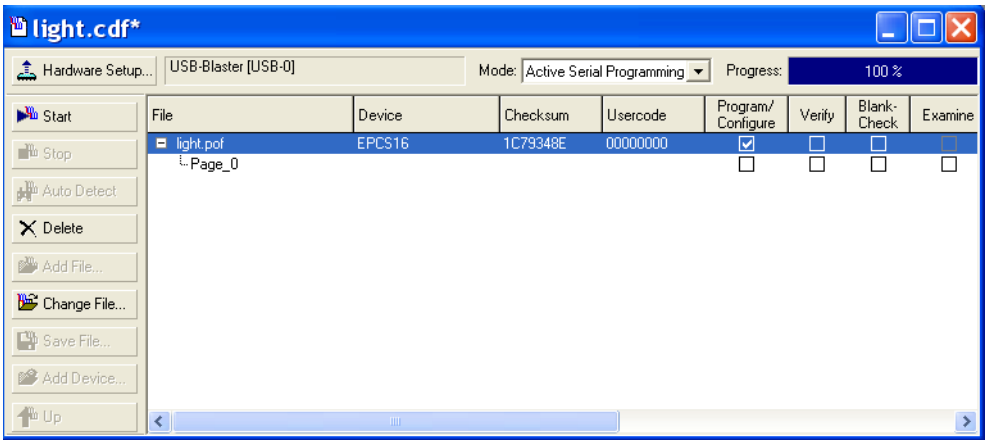


Figure 53. The Programmer window upon completion of programming.

8 Testing the Designed Circuit

Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. Flip the RUN/PROG switch to RUN position. Try all four valuations of the input variables x_1 and x_2, by setting the corresponding states of the switches SW_1 and SW_0.
Write downd the truth table in a paper. Then erify that the circuit implements the truth table in Figure 12.

If you want to make changes in the designed circuit, first close the Programmer window. Then make the desired changes in the Block Diagram/Schematic file, compile the circuit, and program the board as explained above.

Show that your circuit works fine to the TA before leaving the lab.

Table 4.1. Pin assignments for the toggle switches.

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_N25	Toggle Switch[0]
SW[1]	PIN_N26	Toggle Switch[1]
SW[2]	PIN_P25	Toggle Switch[2]
SW[3]	PIN_AE14	Toggle Switch[3]
SW[4]	PIN_AF14	Toggle Switch[4]
SW[5]	PIN_AD13	Toggle Switch[5]
SW[6]	PIN_AC13	Toggle Switch[6]
SW[7]	PIN_C13	Toggle Switch[7]
SW[8]	PIN_B13	Toggle Switch[8]
SW[9]	PIN_A13	Toggle Switch[9]
SW[10]	PIN_N1	Toggle Switch[10]
SW[11]	PIN_P1	Toggle Switch[11]
SW[12]	PIN_P2	Toggle Switch[12]
SW[13]	PIN_T7	Toggle Switch[13]
SW[14]	PIN_U3	Toggle Switch[14]
SW[15]	PIN_U4	Toggle Switch[15]
SW[16]	PIN_V1	Toggle Switch[16]
SW[17]	PIN_V2	Toggle Switch[17]

Table 4.2. Pin assignments for the pushbutton switches.

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_G26	Pushbutton[0]
KEY[1]	PIN_N23	Pushbutton[1]
KEY[2]	PIN_P23	Pushbutton[2]
KEY[3]	PIN_W26	Pushbutton[3]

Table 4.3. Pin assignments for the LEDs.

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_AE23	LED Red[0]
LEDR[1]	PIN_AF23	LED Red[1]
LEDR[2]	PIN_AB21	LED Red[2]
LEDR[3]	PIN_AC22	LED Red[3]
LEDR[4]	PIN_AD22	LED Red[4]
LEDR[5]	PIN_AD23	LED Red[5]
LEDR[6]	PIN_AD21	LED Red[6]
LEDR[7]	PIN_AC21	LED Red[7]
LEDR[8]	PIN_AA14	LED Red[8]
LEDR[9]	PIN_Y13	LED Red[9]
LEDR[10]	PIN_AA13	LED Red[10]
LEDR[11]	PIN_AC14	LED Red[11]
LEDR[12]	PIN_AD15	LED Red[12]
LEDR[13]	PIN_AE15	LED Red[13]
LEDR[14]	PIN_AF13	LED Red[14]
LEDR[15]	PIN_AE13	LED Red[15]
LEDR[16]	PIN_AE12	LED Red[16]
LEDR[17]	PIN_AD12	LED Red[17]
LEDG[0]	PIN_AE22	LED Green[0]
LEDG[1]	PIN_AF22	LED Green[1]
LEDG[2]	PIN_W19	LED Green[2]
LEDG[3]	PIN_V18	LED Green[3]
LEDG[4]	PIN_U18	LED Green[4]
LEDG[5]	PIN_U17	LED Green[5]
LEDG[6]	PIN_AA20	LED Green[6]
LEDG[7]	PIN_Y18	LED Green[7]
LEDG[8]	PIN_Y12	LED Green[8]

4.3 Using the 7-segment Displays

The DE2 Board has eight 7-segment displays. These displays are arranged into two pairs and a group of four, with the intent of displaying numbers of various sizes.

Applying a low logic level to a segment causes it to light up & applying a high logic level turns it off.

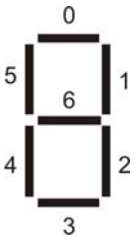


Figure 4.4. Position and index of each segment in a 7-segment display.

Table 4.5. Pin assignments for the 7-segment displays.

Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_AF10	Seven Segment Digit 0[0]
HEX0[1]	PIN_AB12	Seven Segment Digit 0[1]
HEX0[2]	PIN_AC12	Seven Segment Digit 0[2]
HEX0[3]	PIN_AD11	Seven Segment Digit 0[3]
HEX0[4]	PIN_AE11	Seven Segment Digit 0[4]
HEX0[5]	PIN_V14	Seven Segment Digit 0[5]
HEX0[6]	PIN_V13	Seven Segment Digit 0[6]
HEX1[0]	PIN_V20	Seven Segment Digit 1[0]
HEX1[1]	PIN_V21	Seven Segment Digit 1[1]
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]
HEX1[3]	PIN_Y22	Seven Segment Digit 1[3]
HEX1[4]	PIN_AA24	Seven Segment Digit 1[4]
HEX1[5]	PIN_AA23	Seven Segment Digit 1[5]
HEX1[6]	PIN_AB24	Seven Segment Digit 1[6]
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]
HEX2[1]	PIN_V22	Seven Segment Digit 2[1]
HEX2[2]	PIN_AC25	Seven Segment Digit 2[2]
HEX2[3]	PIN_AC26	Seven Segment Digit 2[3]
HEX2[4]	PIN_AB26	Seven Segment Digit 2[4]
HEX2[5]	PIN_AB25	Seven Segment Digit 2[5]
HEX2[6]	PIN_Y24	Seven Segment Digit 2[6]
HEX3[0]	PIN_Y23	Seven Segment Digit 3[0]
HEX3[1]	PIN_AA25	Seven Segment Digit 3[1]
HEX3[2]	PIN_AA26	Seven Segment Digit 3[2]
HEX3[3]	PIN_Y26	Seven Segment Digit 3[3]
HEX3[4]	PIN_Y25	Seven Segment Digit 3[4]
HEX3[5]	PIN_U22	Seven Segment Digit 3[5]
HEX3[6]	PIN_W24	Seven Segment Digit 3[6]
HEX4[0]	PIN_U9	Seven Segment Digit 4[0]
HEX4[1]	PIN_U1	Seven Segment Digit 4[1]
HEX4[2]	PIN_U2	Seven Segment Digit 4[2]
HEX4[3]	PIN_T4	Seven Segment Digit 4[3]
HEX4[4]	PIN_R7	Seven Segment Digit 4[4]
HEX4[5]	PIN_R6	Seven Segment Digit 4[5]
HEX4[6]	PIN_T3	Seven Segment Digit 4[6]

HEX5[0]	PIN_T2	Seven Segment Digit 5[0]
HEX5[1]	PIN_P6	Seven Segment Digit 5[1]
HEX5[2]	PIN_P7	Seven Segment Digit 5[2]
HEX5[3]	PIN_T9	Seven Segment Digit 5[3]
HEX5[4]	PIN_R5	Seven Segment Digit 5[4]
HEX5[5]	PIN_R4	Seven Segment Digit 5[5]
HEX5[6]	PIN_R3	Seven Segment Digit 5[6]
HEX6[0]	PIN_R2	Seven Segment Digit 6[0]
HEX6[1]	PIN_P4	Seven Segment Digit 6[1]
HEX6[2]	PIN_P3	Seven Segment Digit 6[2]
HEX6[3]	PIN_M2	Seven Segment Digit 6[3]
HEX6[4]	PIN_M3	Seven Segment Digit 6[4]
HEX6[5]	PIN_M5	Seven Segment Digit 6[5]
HEX6[6]	PIN_M4	Seven Segment Digit 6[6]
HEX7[0]	PIN_L3	Seven Segment Digit 7[0]
HEX7[1]	PIN_L2	Seven Segment Digit 7[1]
HEX7[2]	PIN_L9	Seven Segment Digit 7[2]
HEX7[3]	PIN_L6	Seven Segment Digit 7[3]
HEX7[4]	PIN_L7	Seven Segment Digit 7[4]
HEX7[5]	PIN_P9	Seven Segment Digit 7[5]
HEX7[6]	PIN_N9	Seven Segment Digit 7[6]