Introduction to Quartus by a VHDL based Design

COE608: Computer Organization and Architecture

Lab # 1: Quartus-II Tutorial

1. Lab Objectives

This tutorial lab has been constructed to introduce the Quartus CAD system from Altera. It provides a general overview of a typical CAD flow for designing digital circuits that are implemented by using Altera FPGA devices. The design process is illustrated by giving step-by-step instructions for using the Quartus II software to implement a very simple circuit in a Cyclone IV FPGA device. This lab-tutorial is a good starting point to the general concept of developing digital circuit in FPGAs. The Quartus II system includes support for a number of methods of entering a description of the desired circuit into (Schematic, Verilog, VHDL, etc.) a CAD system. This tutorial makes use of the VHDL design entry method. Final step in the design process involves configuring the designed circuit in a Cyclone IV FPGA device as part of Altera Development and Education board (DE2-115).

2. Introduction

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

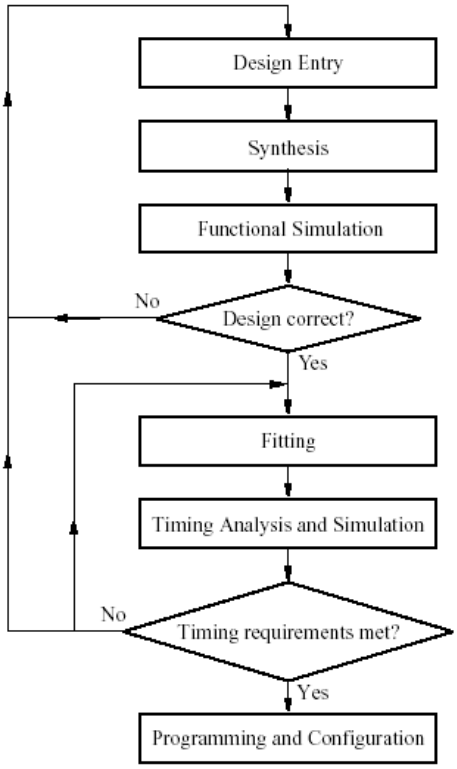


Figure 1: A typical CAD flow

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 using the VHDL hardware description language. It makes use of the

graphical user interface to invoke the Quartus II commands. Doing this tutorial, the students will learn about:

- Creating a project
- Design entry using VHDL code
- Synthesizing a circuit specified in VHDL code
- 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 (Cyclone-IV E) chip on Altera's DE2-115 board

Each logic circuit, or sub-circuit, being designed with Quartus II CAD software is called a *project*. The CAD software works on one project at a time and keeps all information for that project in a single directory. To begin a new logic circuit design, the first step is to create a directory to hold its files. We will use a directory *introtutorial* to hold the design files for this lab. The example circuit for this tutorial is a simple circuit for two-way light control.

Start the Quartus II 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, which the user selects with the computer mouse. Most of the commands provided by Quartus II 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.

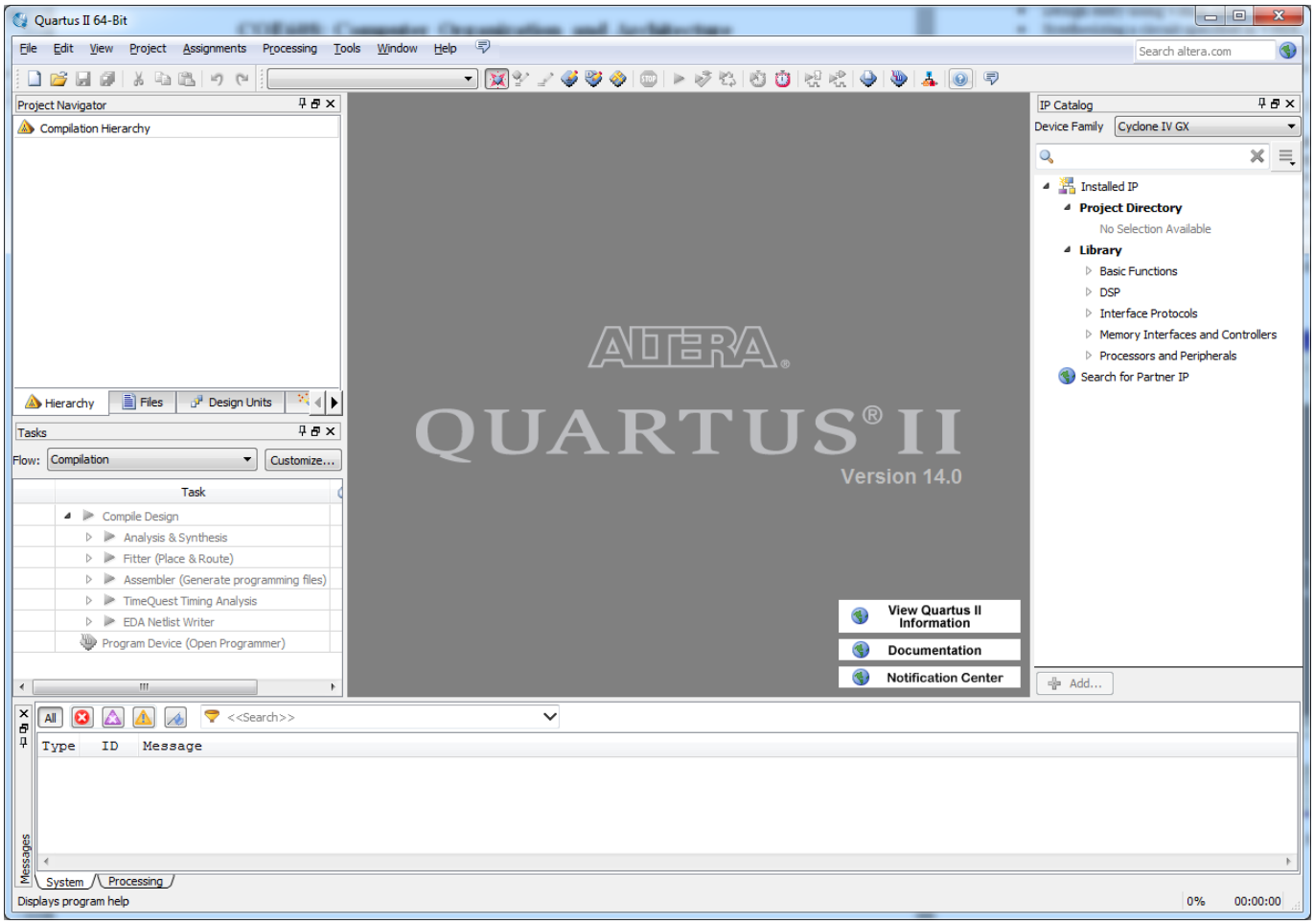


Figure 2. Starting window for Quartus II

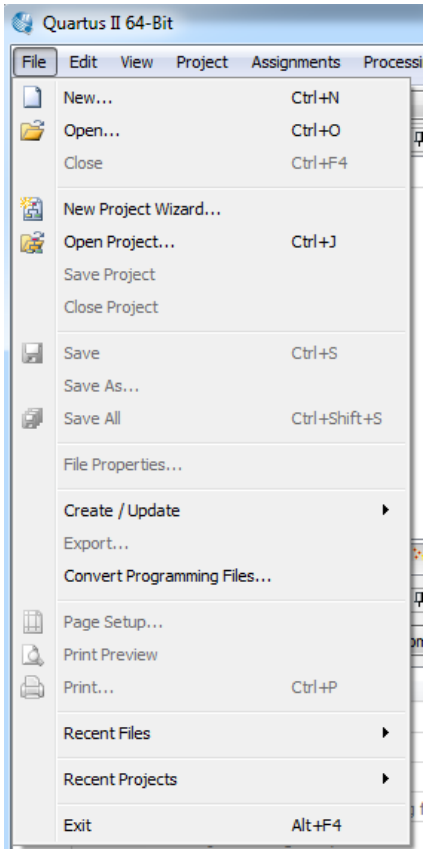


Figure 3. An example of the File menu.

3. 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:

Select File > New Project Wizard to reach the window in Figure 4. Set the working directory to *introtutorial*; you can also use some other directory name of your choice. 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 4. Press Next. Since we have not yet created the directory *introtutorial*, Quartus II software displays the pop-up box asking if it should create the desired directory. Click Yes, which leads to the window in Figure 5.

i) 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 6.

ii) We have to specify the type of device in which the designed circuit will be implemented. Choose Cyclone-IV E as the target device family. From the list of available devices, choose the device called EP4CE115F29C7 that is the FPGA used on Altera's DE2-115 board. Press Next, which opens another window. The user can specify any third-party tools (*EDA tools)* that should be used. Since we will rely solely on Quartus II tools, we will not choose any other tools. Therefore, choose NONE for all fields and press Next.

iii) A summary of the chosen settings appears in the screen shown in Figure 7. Press Finish, which returns to the main Quartus window, but with *light* specified as project, in the display title bar, as shown in Figure 8.

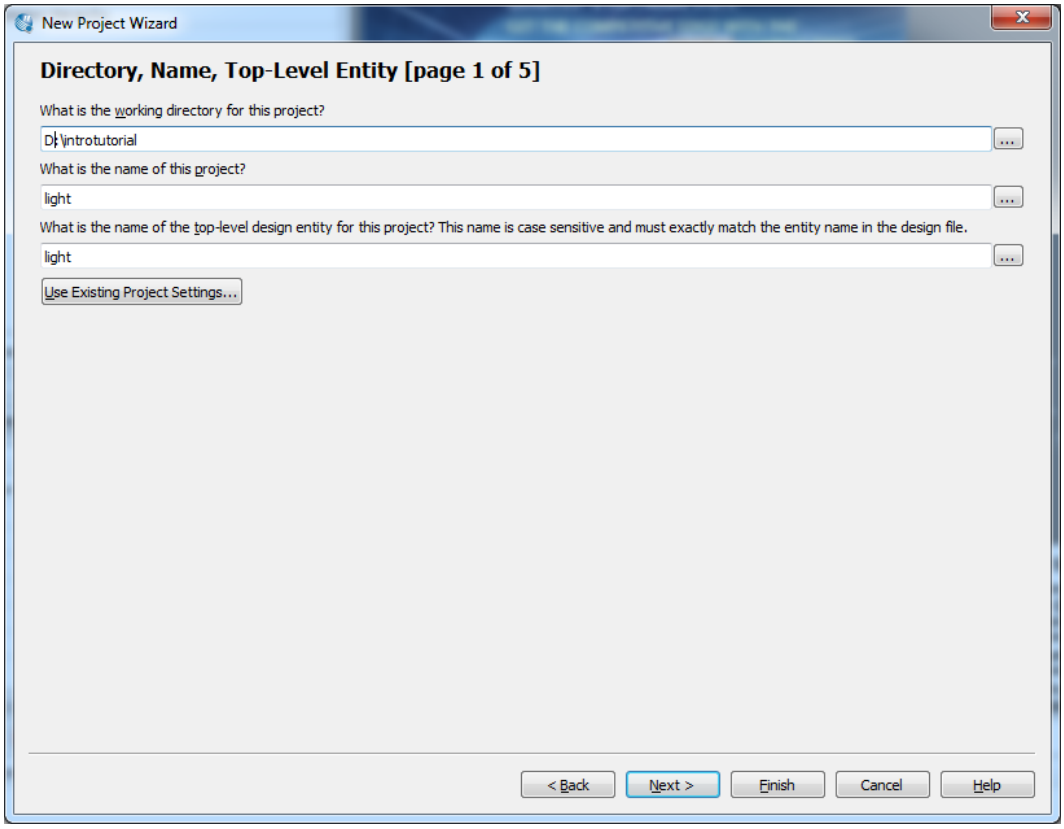


Figure 4. Creation of a new project.

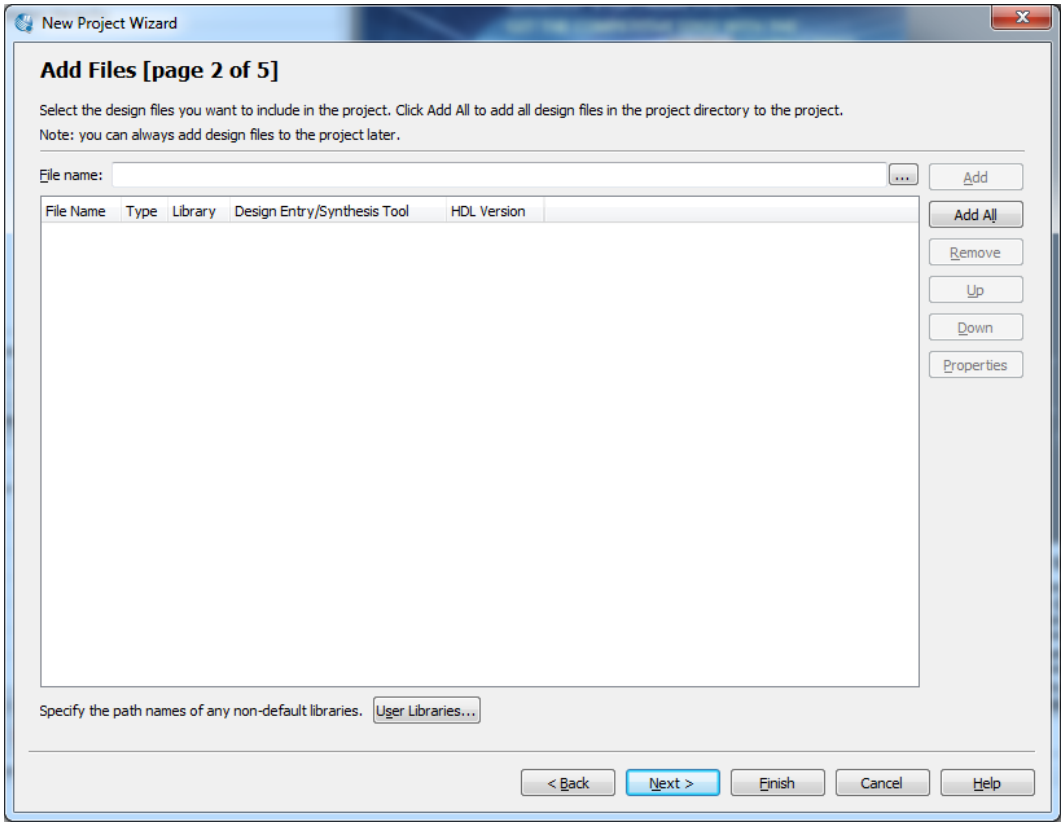


Figure 5. The wizard can include user-specified design files.

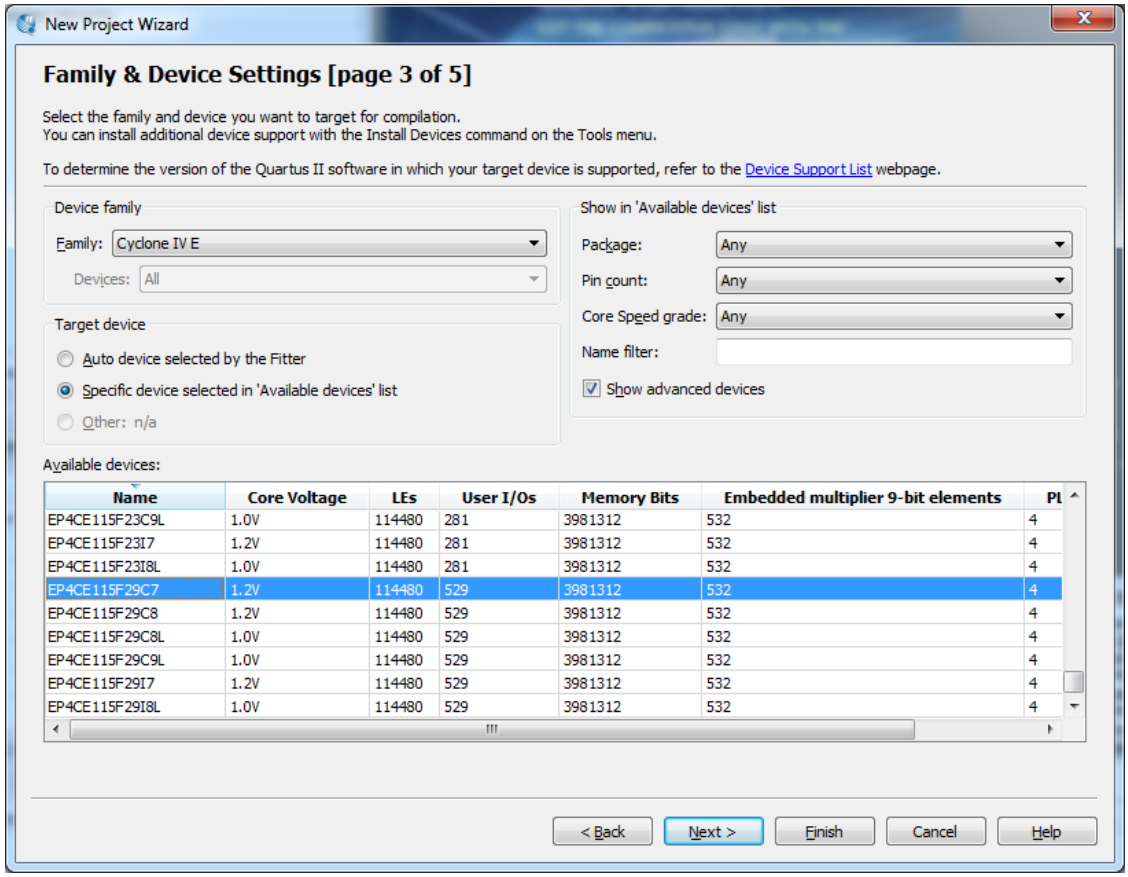


Figure 6. Choose the device family and a specific device.

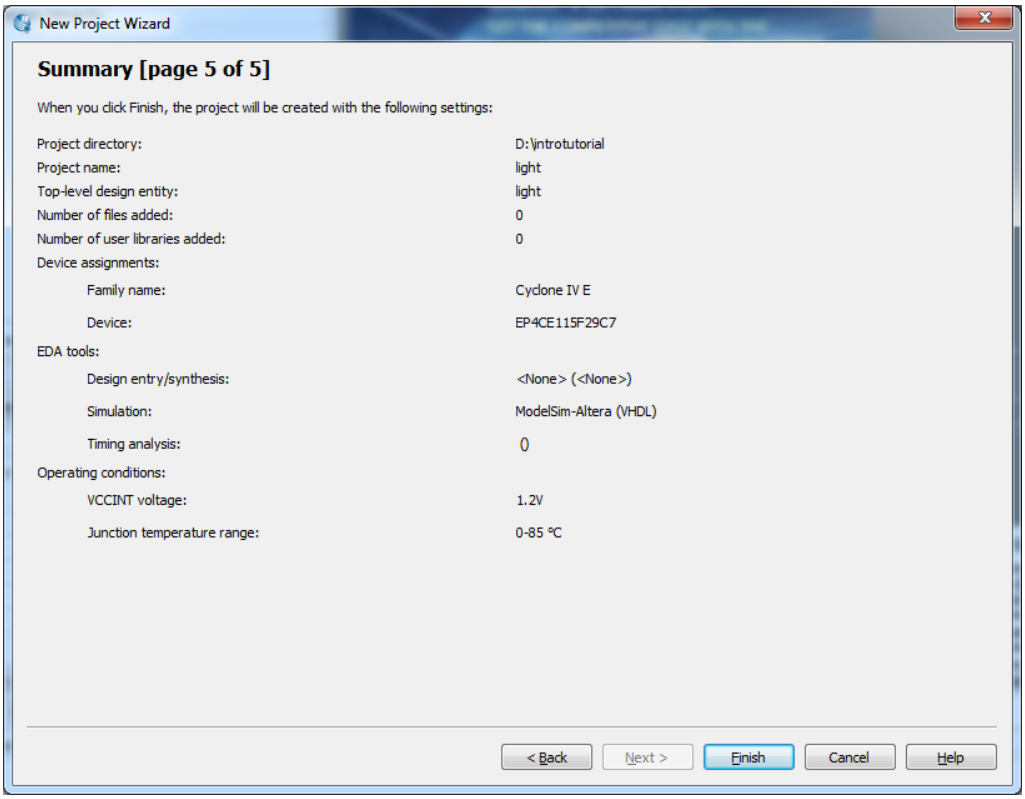


Figure 7. Summary of the project settings.

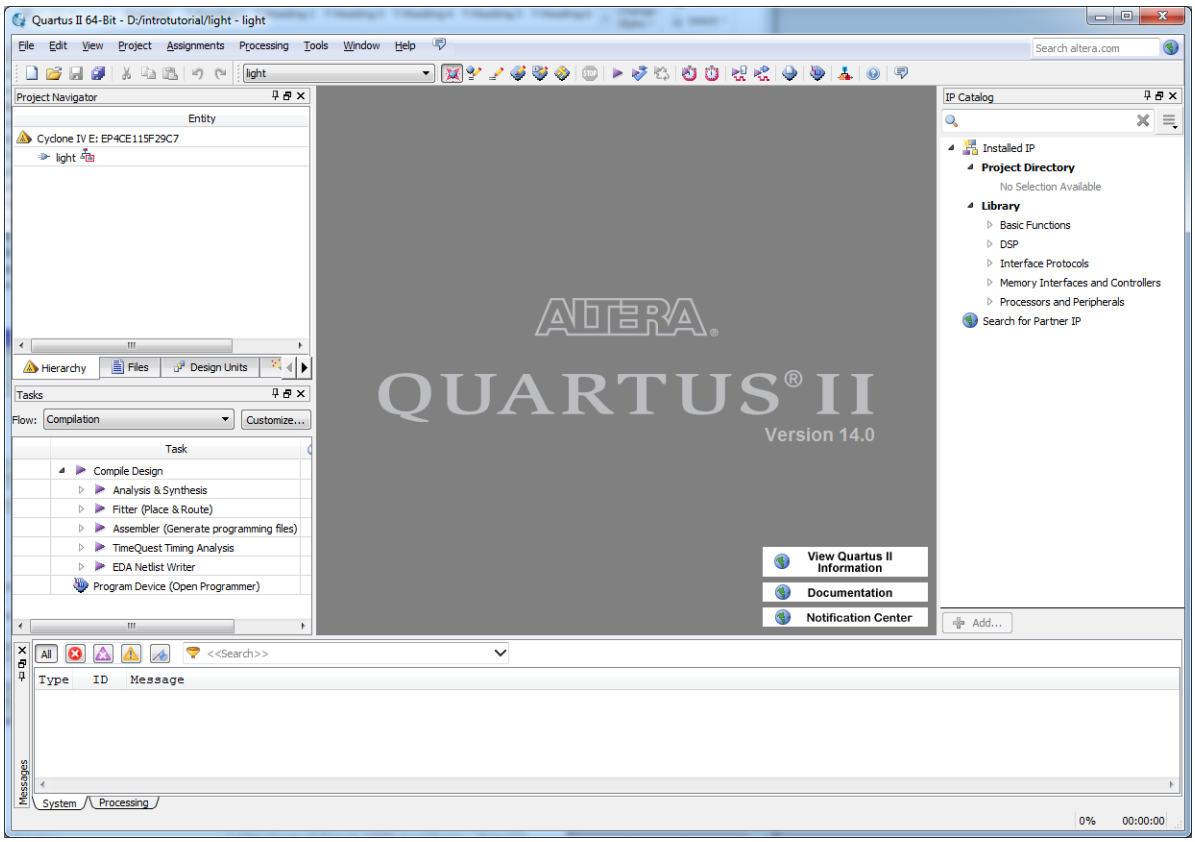


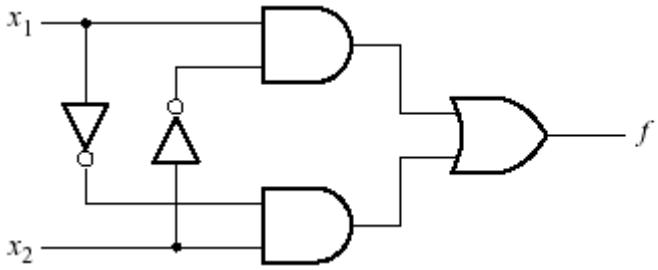
Figure 8. The Quartus II display for the created project.

4. Design Entry Using VHDL Code

As a design example, we use the two-way light controller circuit shown in Figure 10. 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. This is just an Exclusive-OR function of the inputs x_1 and x_2, but we will specify it using the gates shown in Figure 9.

The required circuit is described by the VHDL code in Figure 10. The VHDL entity is called *light* to match the name given in Figure 4. This code can be typed into a file by using any text editor that stores ASCII files, or by using the Quartus II text editing facilities. While the file can be given any name, it is a common designers' practice to use the same name as the name of the top-level VHDL entity. The file name must include the extension vhd, which indicates a VHDL file. So, we use the name *light.vhd* and use the Quartus II Text Editor to create the VHDL source code file *light.vhd*. To help with syntax of the VHDL code, the Quartus Text Editor provides a collection of *VHDL templates*. The templates provide examples of various types of VHDL statements, such as an ENTITY declaration, a CASE statement, and assignment statements. It is worthwhile to browse through the templates by selecting Edit > Insert Template > VHDL to become familiar with this resource.

Select File > New, choose VHDL File, and click OK. This opens the Text Editor window. The first step is to specify a name for the file that will be created. Select File > Save As. In the box labeled Save as type choose VHDL File. In the box labeled File name type *light*. Put a checkmark in the box Add file to current project. Click Save, which puts the file into the directory *introtutorial*. Maximize the Text Editor window and enter the VHDL code in Figure 10 into it. Save the file by typing File > Save, or by typing the shortcut Ctrl-s.



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

Figure 9. The light controller circuit.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY light IS
  PORT ( x1, x2 : IN STD_LOGIC ;
  f : OUT STD_LOGIC ) ;
END light ;
ARCHITECTURE LogicFunction OF light IS
BEGIN
  f <= (x1 AND NOT x2) OR (NOT x1 AND x2);
END LogicFunction ;
```

Figure 10. VHDL code for the circuit in Figure 9.

Adding Design Files to a Project

As indicated earlier, we can tell the Quartus II software which design files it should use as part of the current project. To see the list of files already included in the *light* project, select Assignments > Settings, which leads to the window in Figure 11. As indicated on the left side of the figure, click on the item Files. An alternative way of making this selection is to choose Project > Add/Remove Files in Project.

If we have used the Quartus II Text Editor to create the file and checked the box labeled Add file to current project, then the *light.vhd* file is already part of the project and will be listed in the window in Figure 11. Otherwise, the file must be added to the project. So, place a copy of the file *light.vhd* into the directory *introtutorial*. To add this file to the project, click on the File name: button in Figure 11 to get the pop-up window in Figure 12. Select the *light.vhd* file and click Open. The selected file is now indicated in the Files window of Figure 11. Click OK to include the *light.vhd* file in the project.

5. Compiling the Designed Circuit

The VHDL code in the file *light.vhd* is processed by several Quartus II tools that analyze the code, synthesize the circuit, and generate an implementation of it for the target FPGA chip. These tools are controlled by the application program *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 13

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. The window can be opened at any time either by selecting Processing > Compilation Report or by clicking on the icon . Figure 13 displays the Compiler Flow Summary section, which indicates that only one logic element and 3 pins are needed to implement this circuit on the selected FPGA.

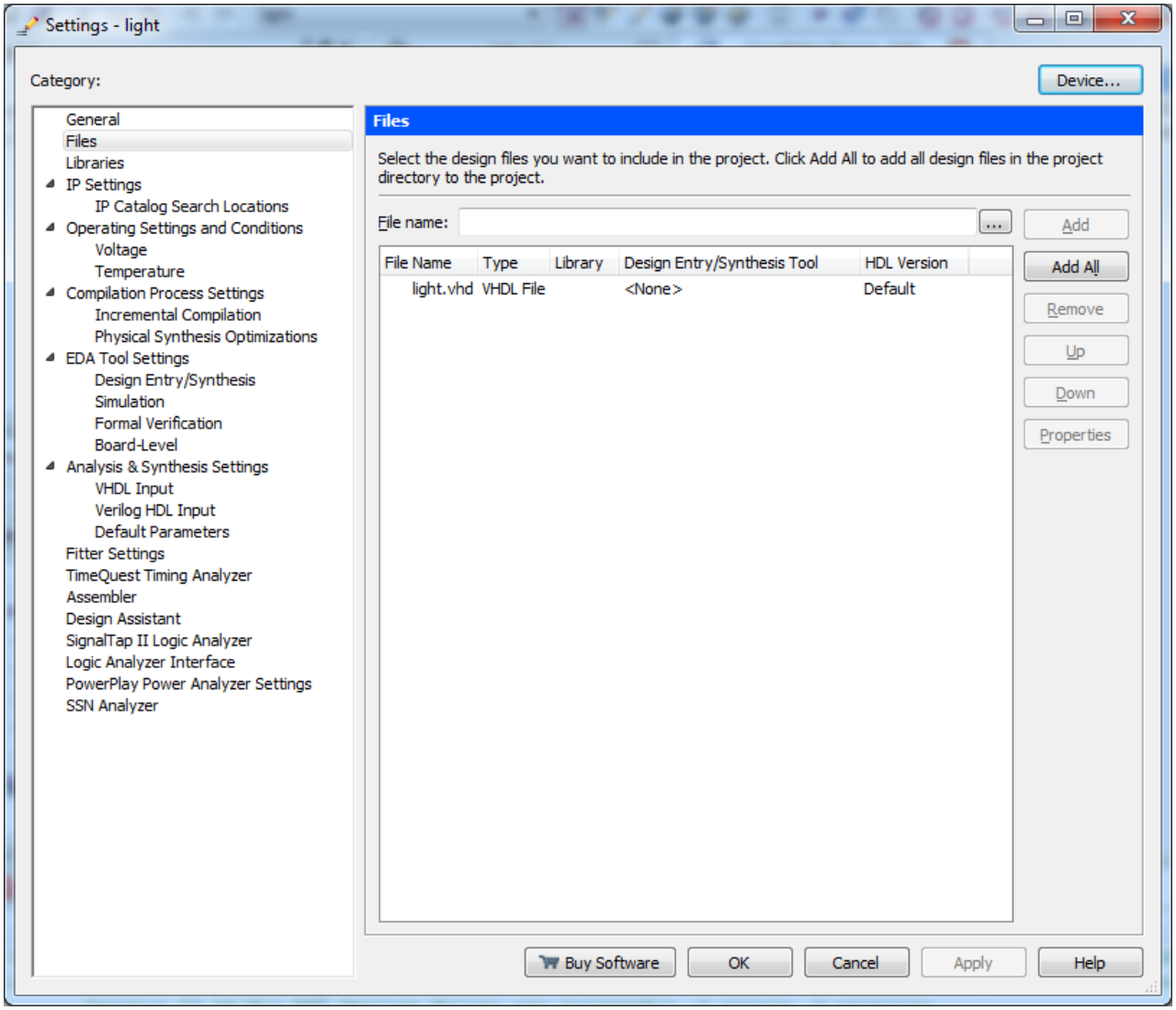


Figure 11. Settings window.

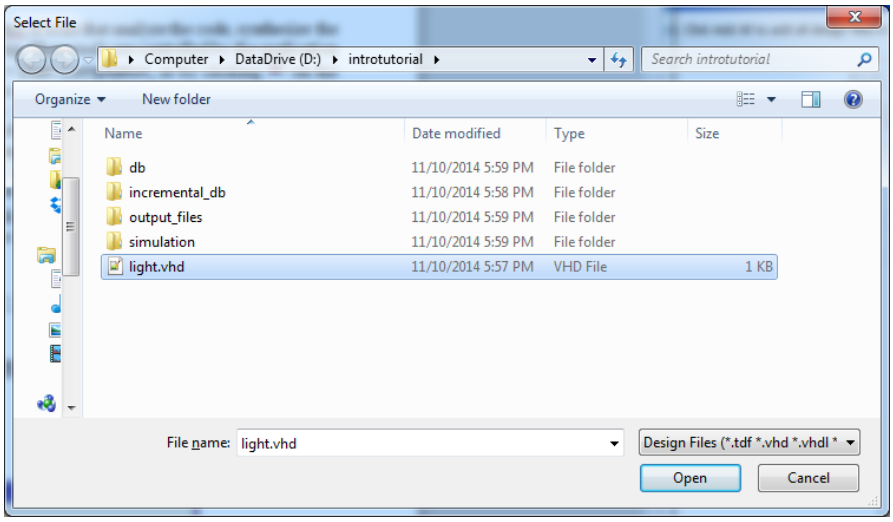


Figure 12. Select the file.

Errors

Quartus II displays messages produced during compilation in the Messages window. If the VHDL design file is correct, one of the messages will state that the compilation is successful and there are no errors. If the Compiler does not report zero errors, then there is at least one mistake in the VHDL code. 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 statement in the VHDL code in the Text Editor window. Similarly, the Compiler may display some warning messages. Their details can be explored in the same way. You can obtain more information about a specific error or warning message by selecting the message and pressing the F1 function key.

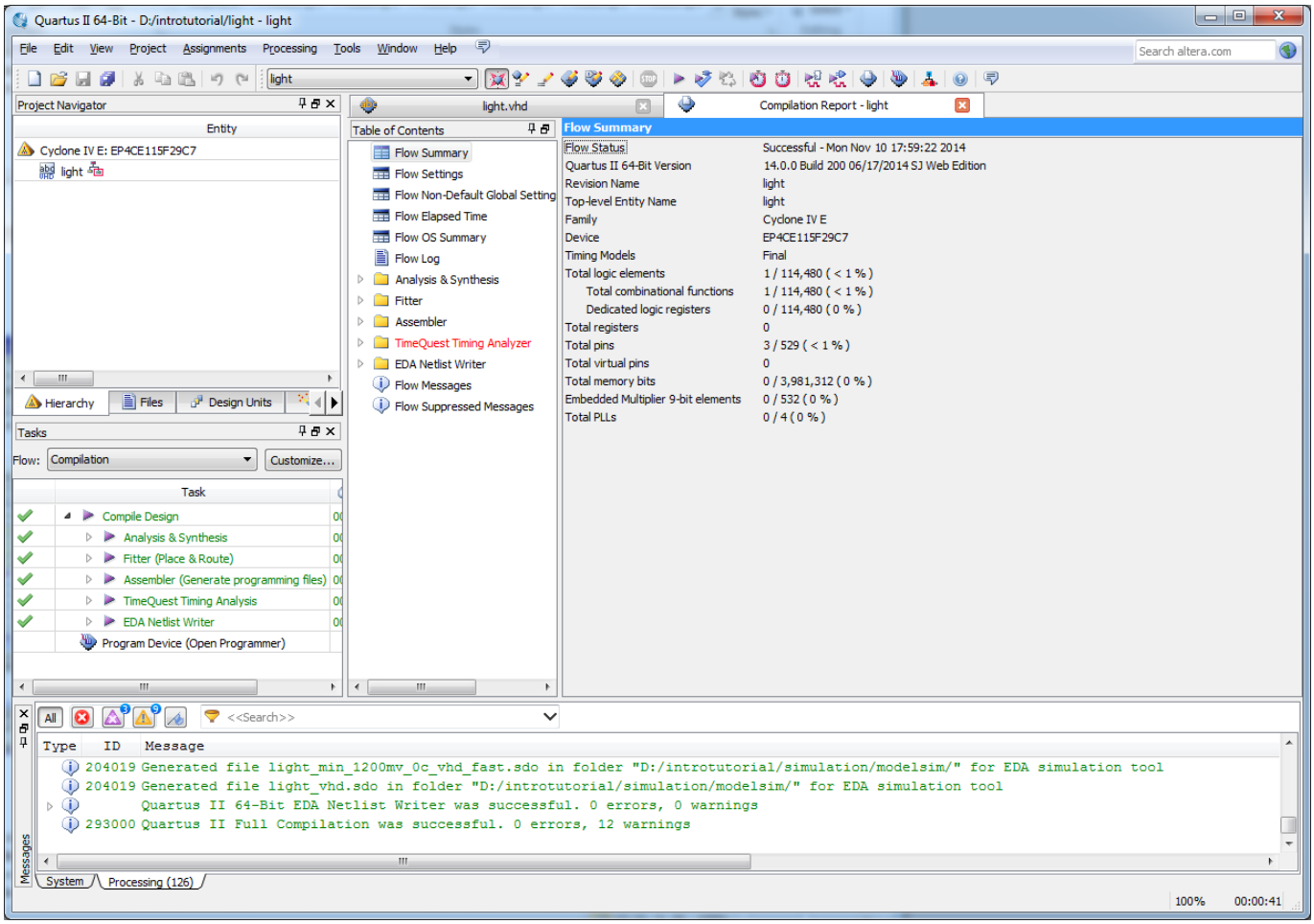


Figure 13. Display after a successful compilation.

To see the effect of an error, open the file *light.vhd*. Remove the semicolon in the statement that defines the function ***f***, illustrating a typographical error that is easily made. Compile the erroneous design file by clicking on the icon . A pop-up box will ask if the changes made to the *light.vhd* 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 14, 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 15. Double-click on the first error message. Quartus II software responds by opening the *light.vhd* file and highlighting the statement, which is affected by the error, as shown in Figure 16. Correct the error and recompile the design.

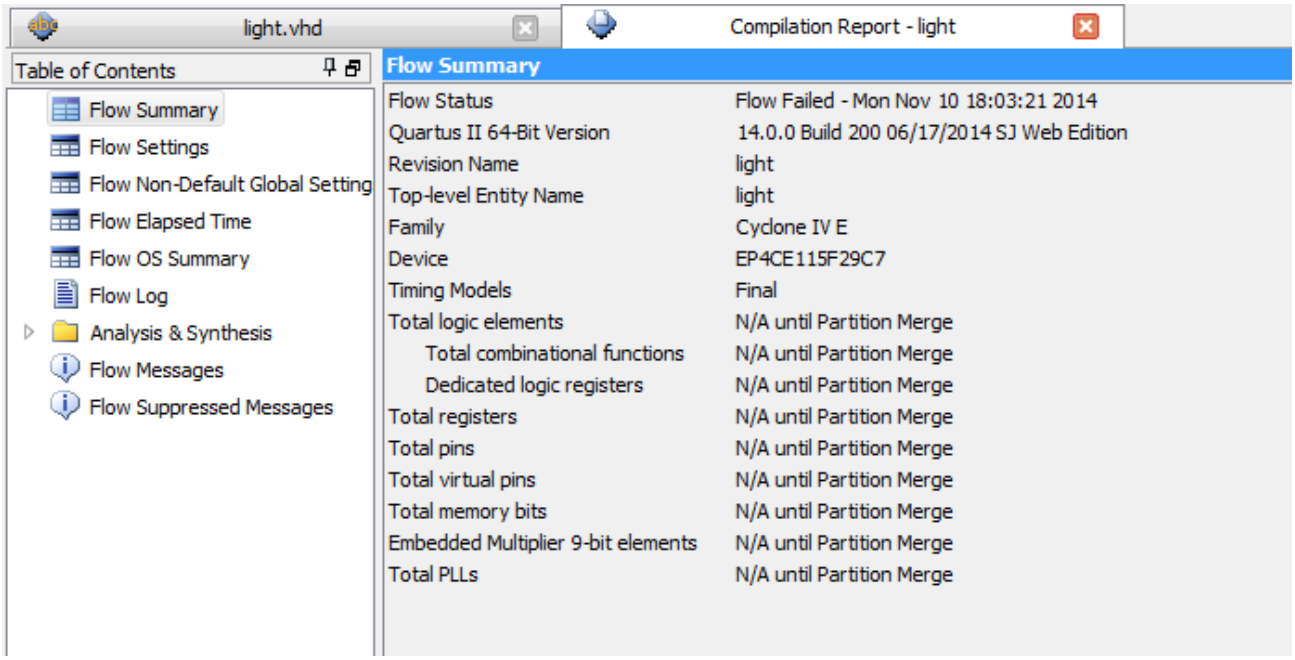


Figure 14. Compilation report for the failed design.

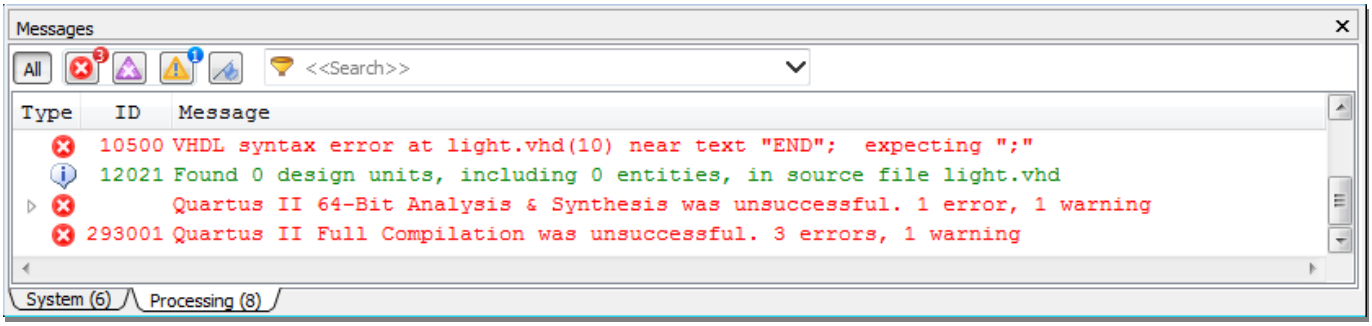


Figure 15. Error messages.

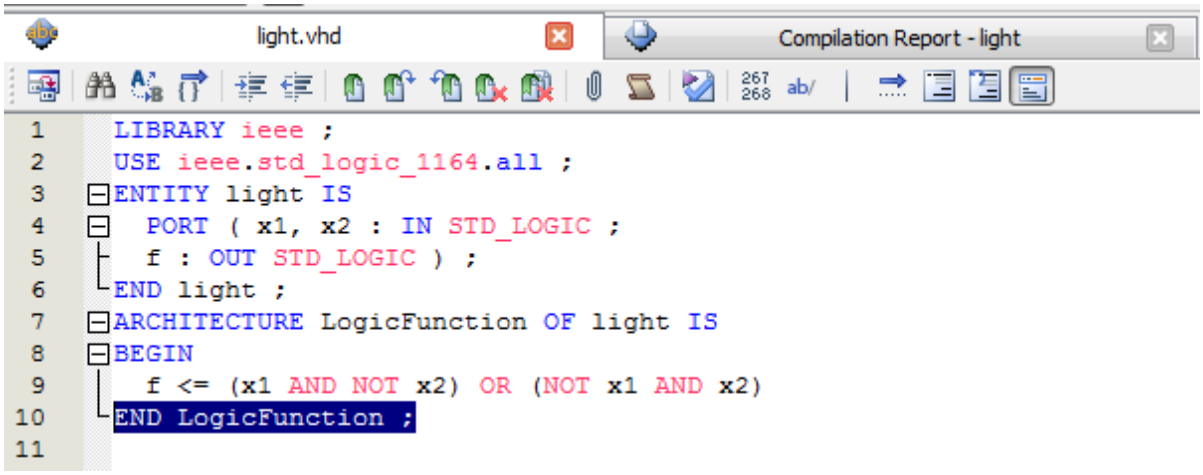


Figure 16. Identifying the location of the error.

6. Pin Assignment

Quartus II Compiler is free to choose any pins on the selected FPGA to serve as inputs and outputs during the compilation. However, the DE2-115 board has hardwired connections between the FPGA pins and the other components on the board. We use two toggle switches, labeled SW1 and SW0, to provide the external inputs, ***x1*** and ***x2***, to our example circuit. These switches are connected to the FPGA pins AB28 and AC28, respectively. We will connect the output ***f*** to the green light-emitting diode labeled LEDG0, which is hardwired to the FPGA pin E21. Pin assignments are made by using the Assignment Editor. Double-click on the entry <<new>> which is highlighted in blue in the column labeled To. Click on the box and then click on the sunglasses shown in Figure 17. The Node Finder Window allows Nodes in the design to be selected. Click List to list all pins, select all nodes on the left hand side and click the > button to select all the nodes would result in Figure 18. Alternatively, manually typing ***x1*** would assign the switch (SW0) as the first pin. In Assignment Name Box, in the drop down menu in Figure 19 find Location (Accepts wildcards/groups). In the Value box, type in PIN_AB28 as this is the pin location hardwired from the FPGA to the switch (SW0). Use the same procedure to assign input ***x2*** to pin AC28 and output ***f*** to pin E21, which results in the image in Figure 20. To save the assignments, 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.

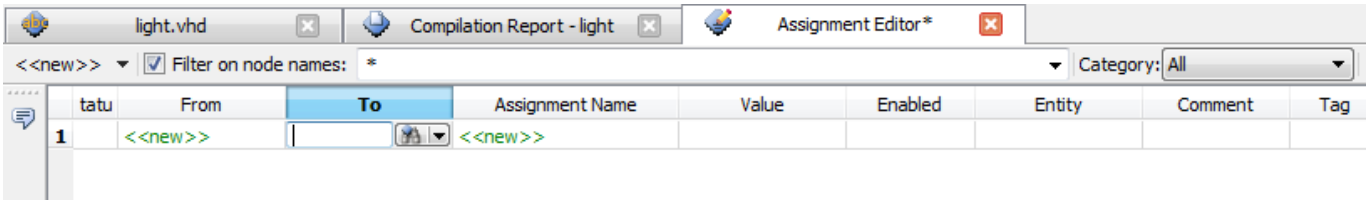


Figure 17. The Assignment Editor window.

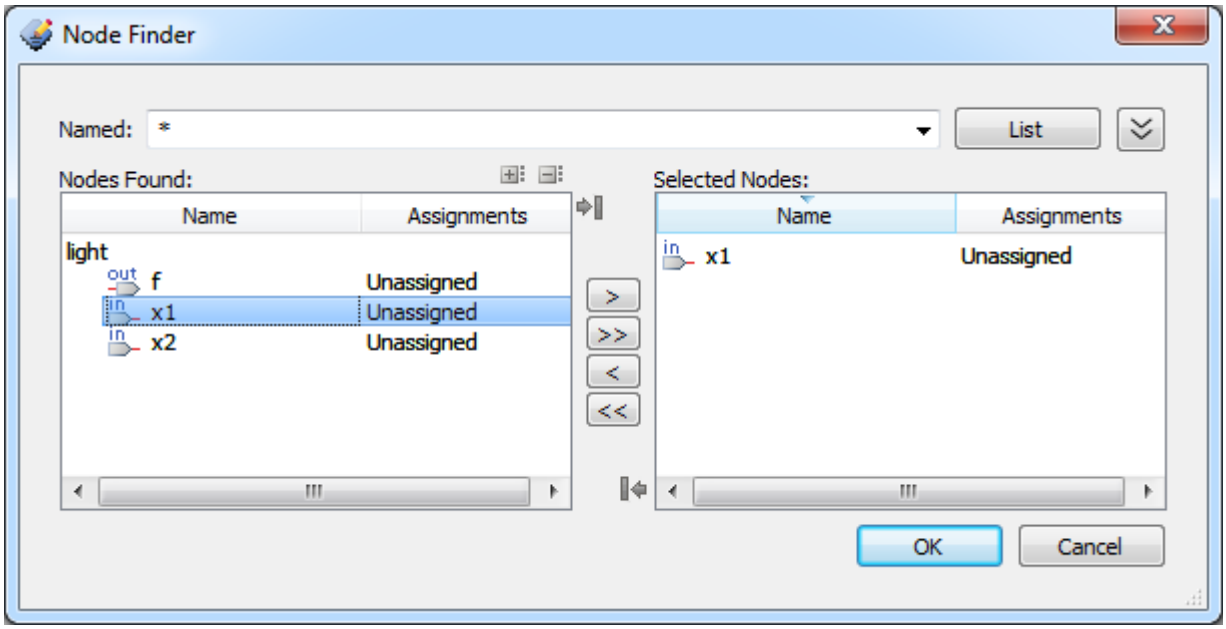


Figure 18. Selecting Assignment Name.

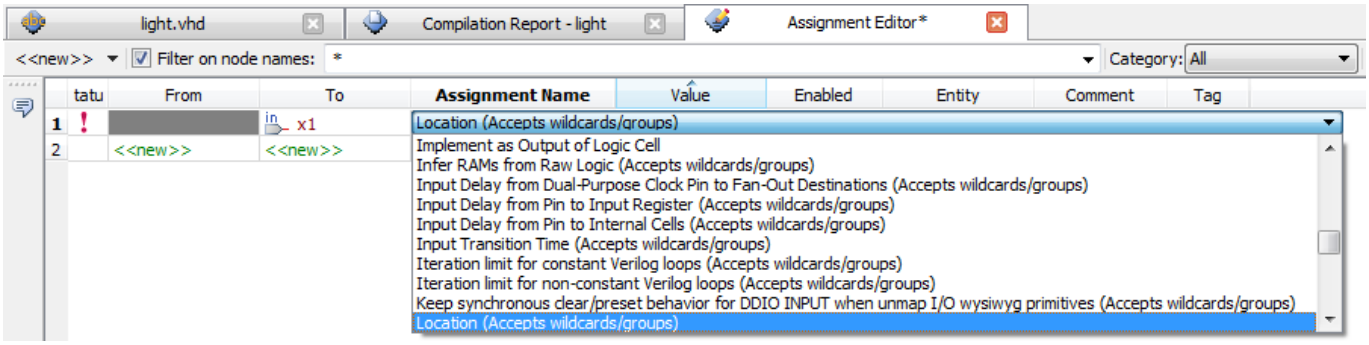


Figure 19. Selecting Assignment Name.

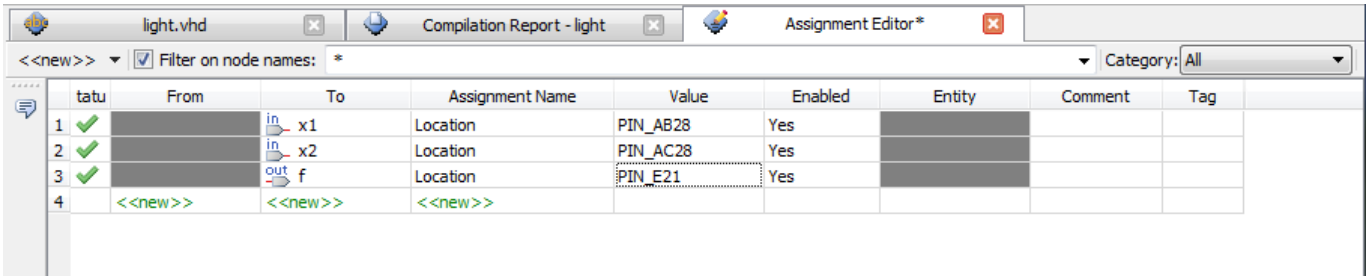


Figure 20. The complete assignment.

The DE2-115 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. If you have 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. Now, select File > Export which leads to the window in Figure 21. Here, the file *light.qsf* is available for export. Click on Export. If you now look in the directory *introtutorial*, you will see that the file *light.qsf* has been created.

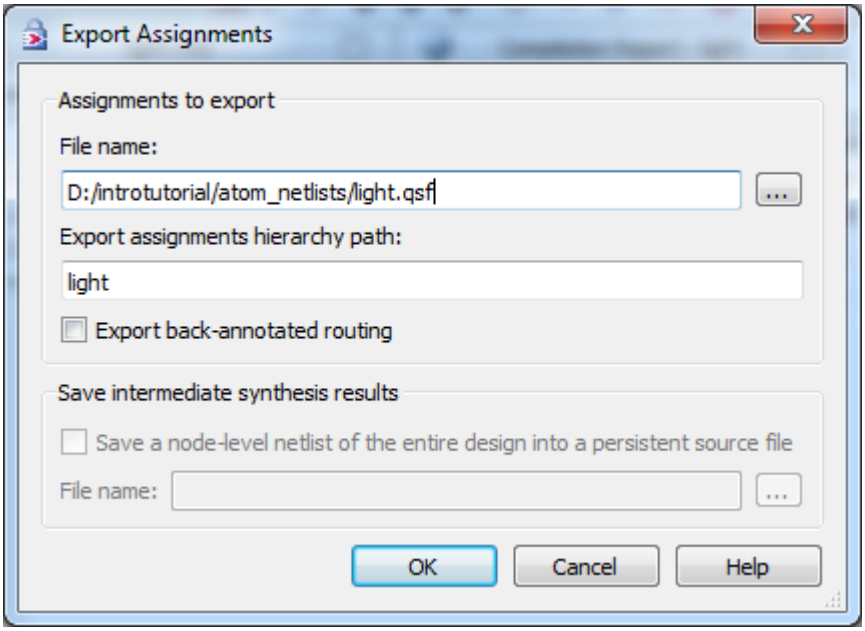


Figure 21. Exporting the pin assignment.

We can import a pin assignment by choosing Assignments > Import Assignments. This opens the dialogue in Figure 22 to select the file to import. Type the name of the file, including the *qsf* 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. For convenience, all relevant pin assignments for the DE2-115 board are given in the file called *DE2_115_pin_assignment.qsf* in the course web directory, **../labs/**, which can also be found on Altera's DE2-115 web pages.

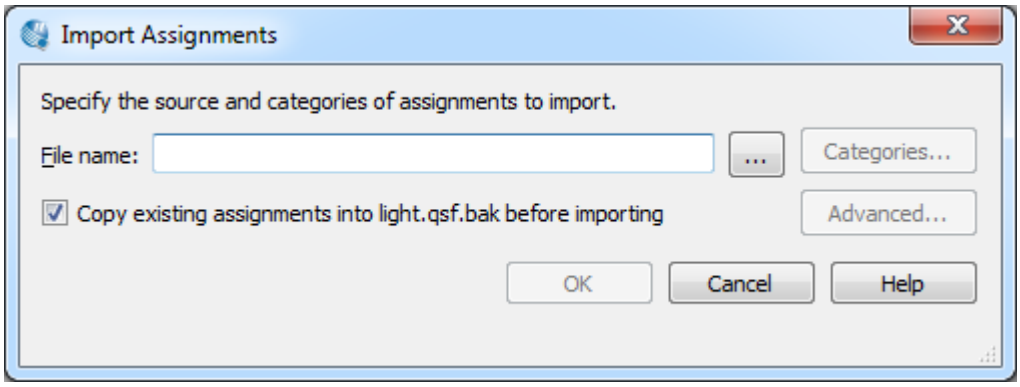


Figure 22. Importing the pin assignment.

7. Simulating the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE2 board, it is prudent to simulate it to verify 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:

i). Open the Waveform Editor window by selecting File > New, which gives the window shown in Figure 23. Choose University Program WVF and click OK.

ii). The Waveform Editor window is depicted in Figure 24. 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 25. You may wish to resize the window to its maximum size.

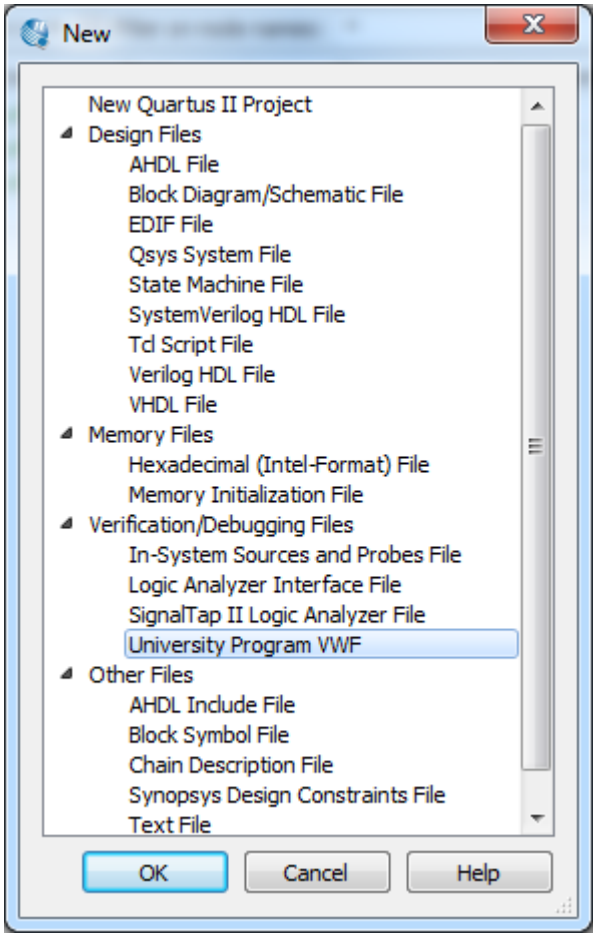


Figure 23. Need to prepare a new file.

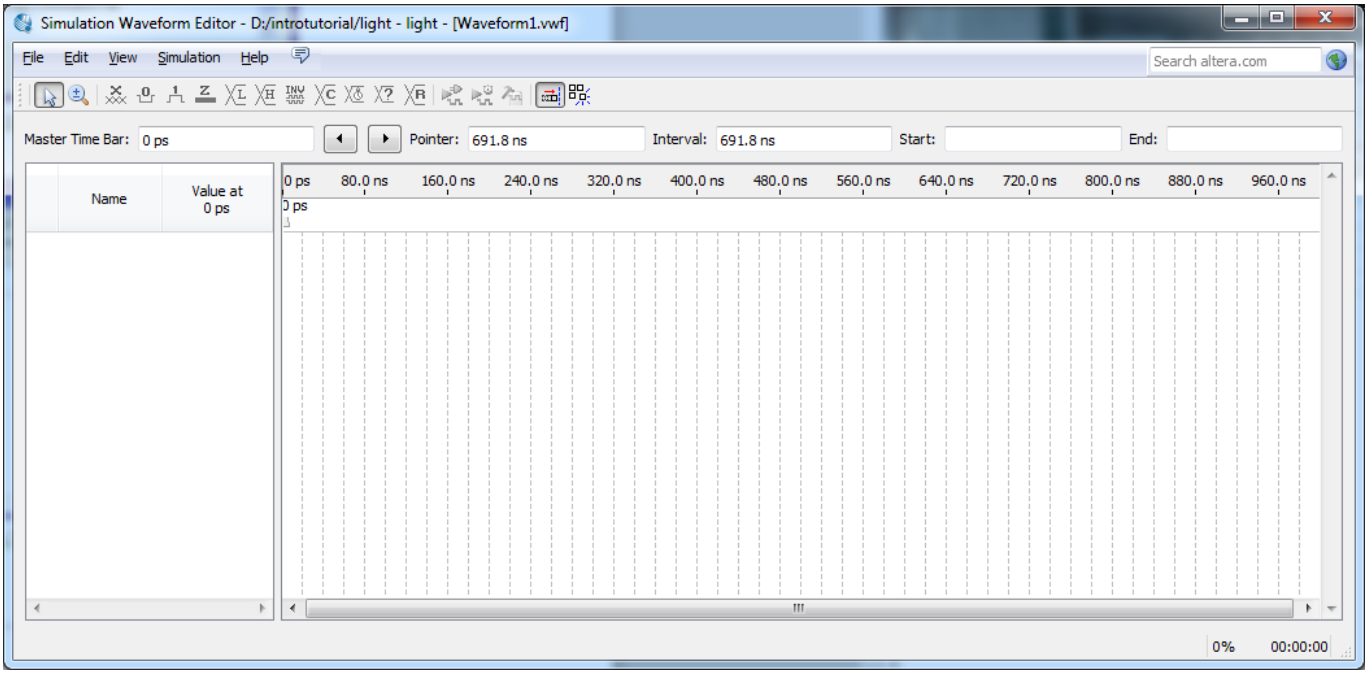


Figure 24. The Waveform Editor window.

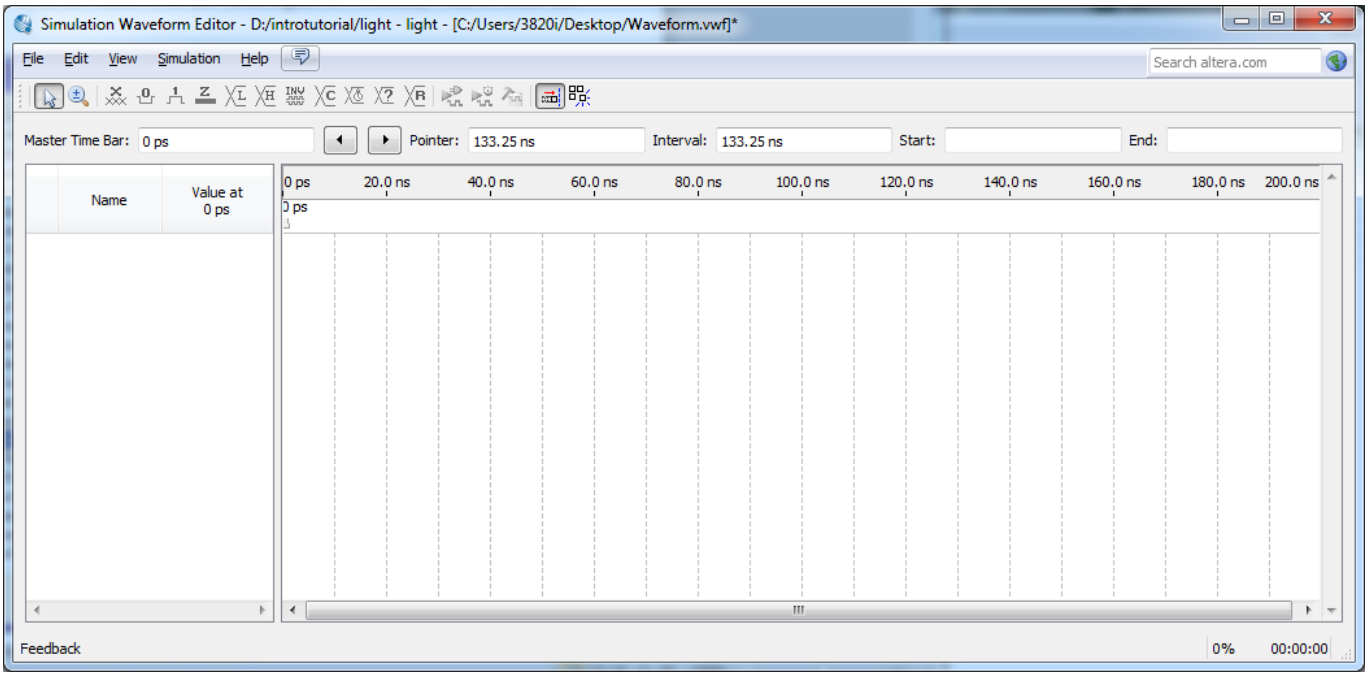


Figure 25. The augmented Waveform Editor window.

iii). Next, we want to include the input and output nodes of the circuit to be simulated. Click Edit > Insert > Insert Node or Bus to open the window in Figure 26. 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 27. The Node Finder utility has a filter used to indicate what types 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. Click on the ***x1*** signal in the Nodes Found box in Figure 27, 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 26. This leaves a fully displayed Waveform Editor window, as shown in Figure 28. If you did not select the nodes in the same order as displayed in Figure 28, 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.

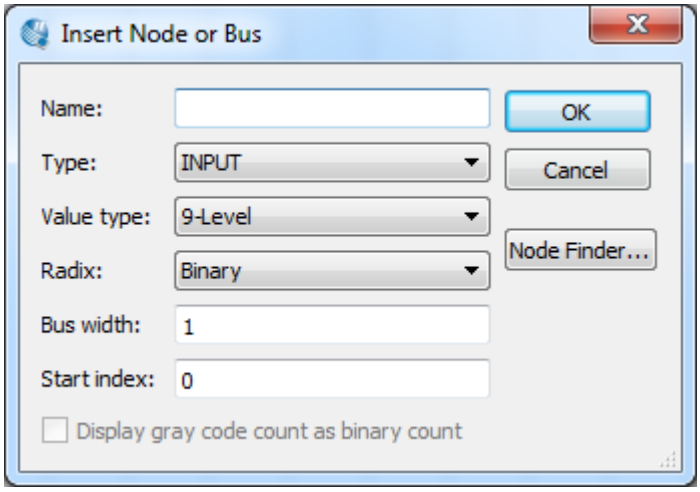


Figure 26. The Insert Node or Bus dialogue.

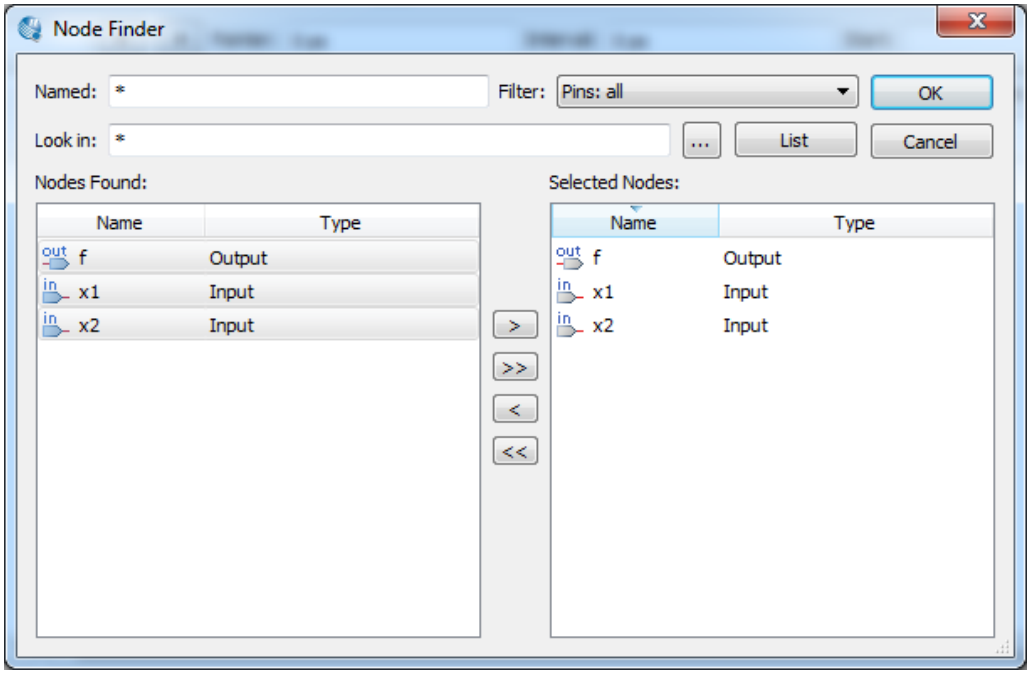


Figure 27. Selecting nodes to insert into the Waveform Editor.

iv). We 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 Edit > 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 .

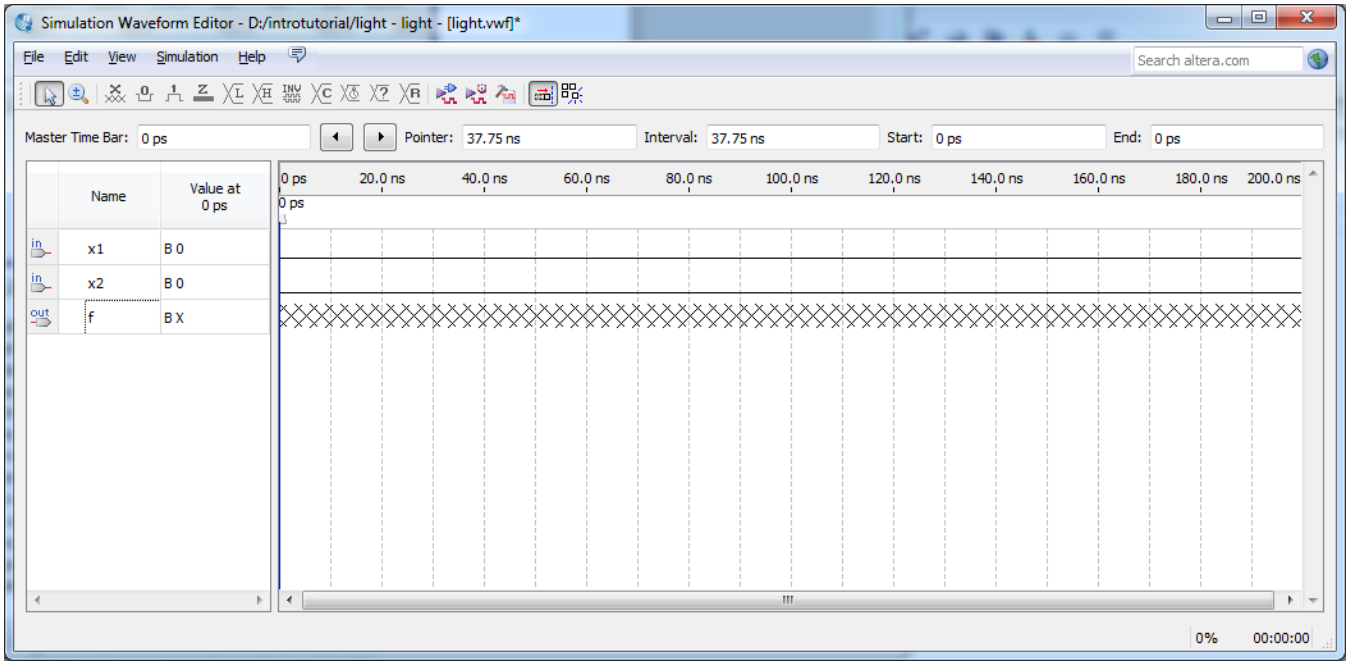


Figure 28. The nodes needed for simulation.

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 9. 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 9. This should produce the image in Figure 29. 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.

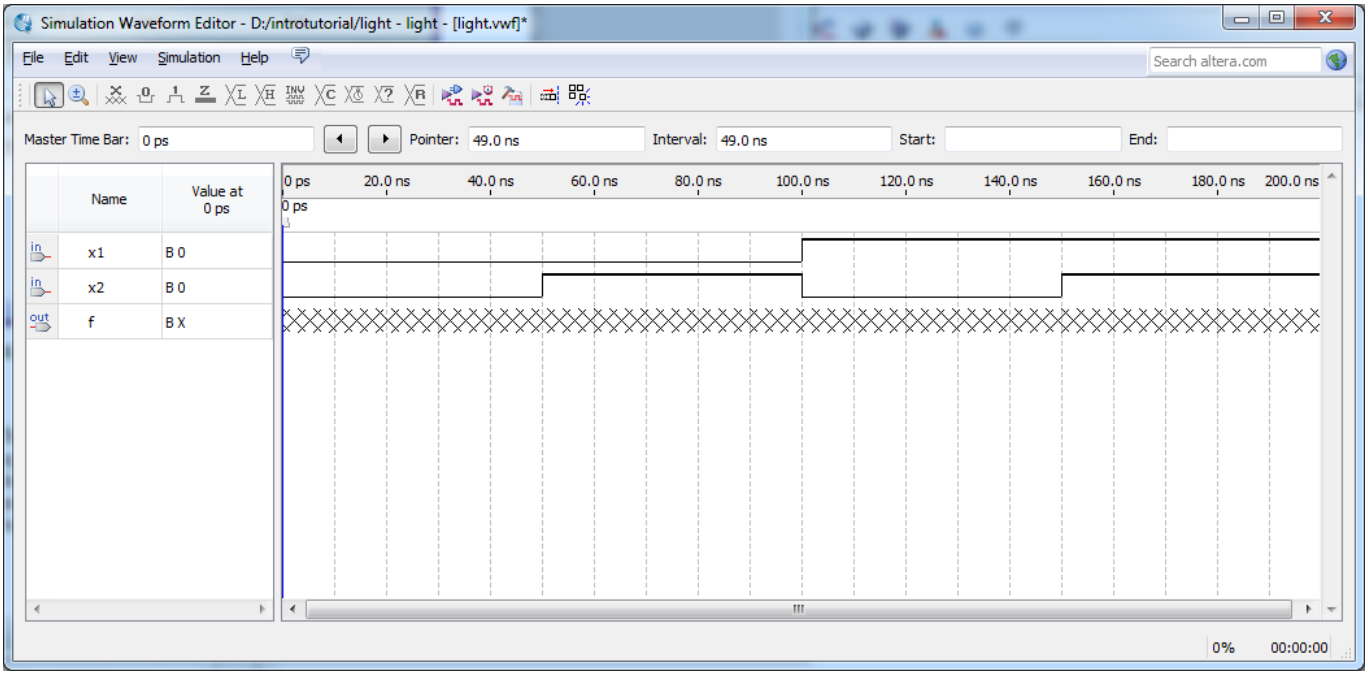


Figure 29. Setting of test values.

7.1 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.

7.2 Functional Simulation

To perform the functional simulation, click on the Run Functional Simulation icon. A Simulation Flow Window will briefly appear as it processes the request to create the simulation waveform. At the end of the simulation, Quartus II software indicates its successful completion and displays the simulated waveform in a new window illustrated in Figure 30. If your waveform 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 10.

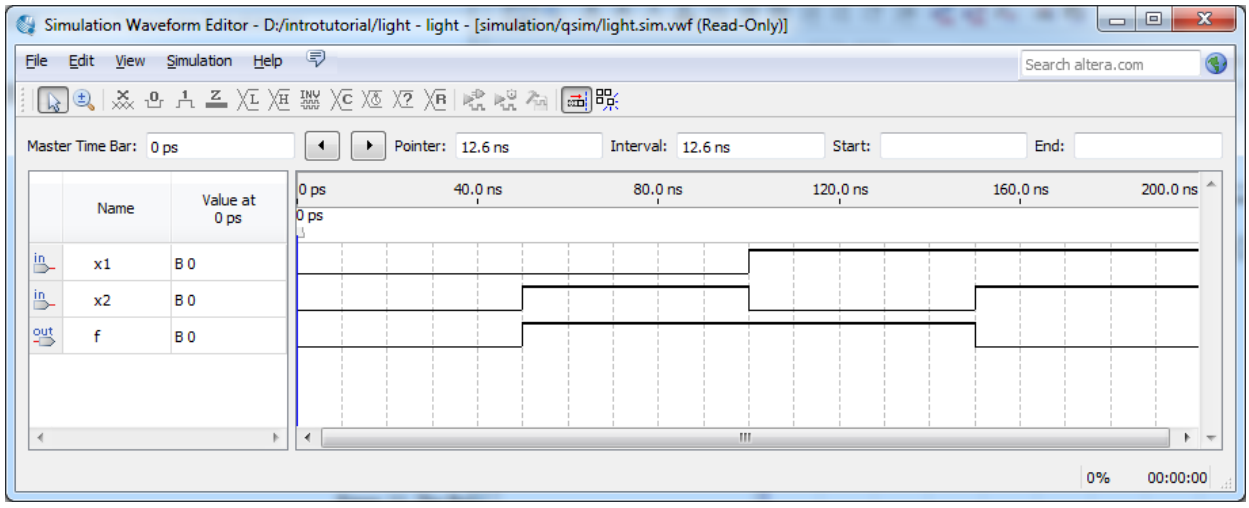


Figure 30. The result of functional simulation.

7.3 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. To perform timing simulation, click on the Run Timing Simulation button which should produce the waveforms in Figure 31. 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, ***x1*** and ***x2***, 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 ***x1*** and ***x2*** may not arrive at exactly the same time at the logic element that generates ***f***.

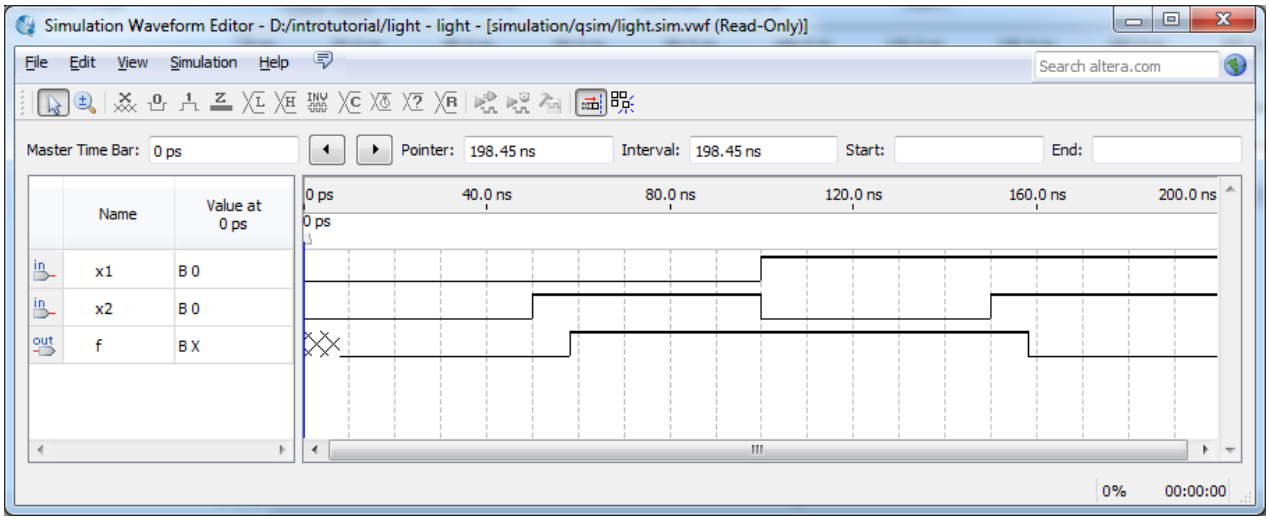


Figure 31. The result of timing simulation.

8. 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 DE2-115 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 that has been already installed. 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 DE2-115 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 DE2-115 board. The RUN position selects the JTAG mode.

JTAG Programming

Program and configure the Cyclone IV E FPGA in the JTAG programming mode as following:

1. Turn on the power to the DE2-115 board. Ensure that the RUN/PROG switch is in the RUN position.
2. Select Tools > Programmer to reach the window in Figure 32.
3. 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, depending on your computer system hardware, as shown in Figure 33.
4. The configuration file *lights.sof* should be listed in the window. The extension *.sof* stands for SRAM Object File. If the file is not already listed, then click Add File and select it (The file should be located inside the output_files folder). Note also that the device selected is EP4CE115F29C7, which is the FPGA device used on the DE2-115 board.
5. Checkmark the box under Program/Configure to select this action (If it is not selected).
6. At this point the window settings should appear as indicated in Figure 34. Press Start to configure the FPGA. An LED on the board will light up when the configuration data has been downloaded successfully.

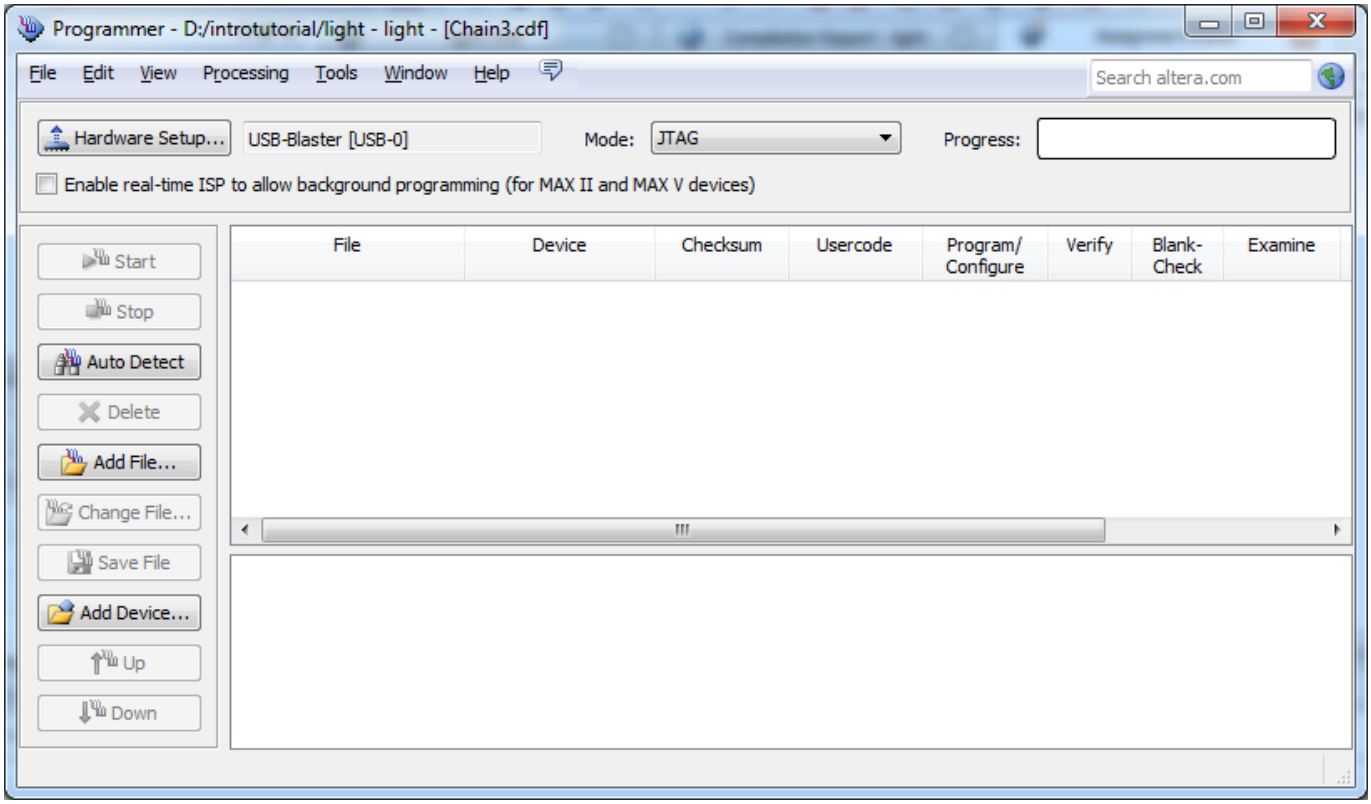


Figure 32. The Programmer window.

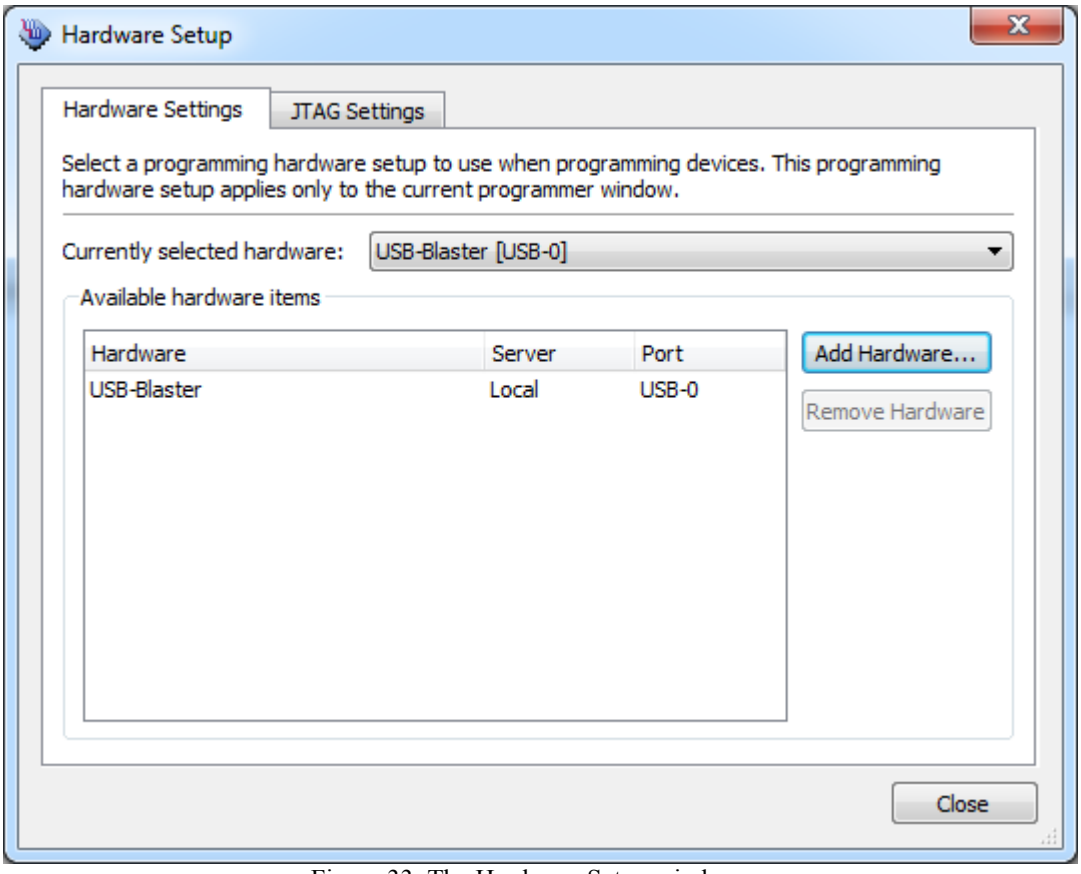


Figure 33. The Hardware Setup window.

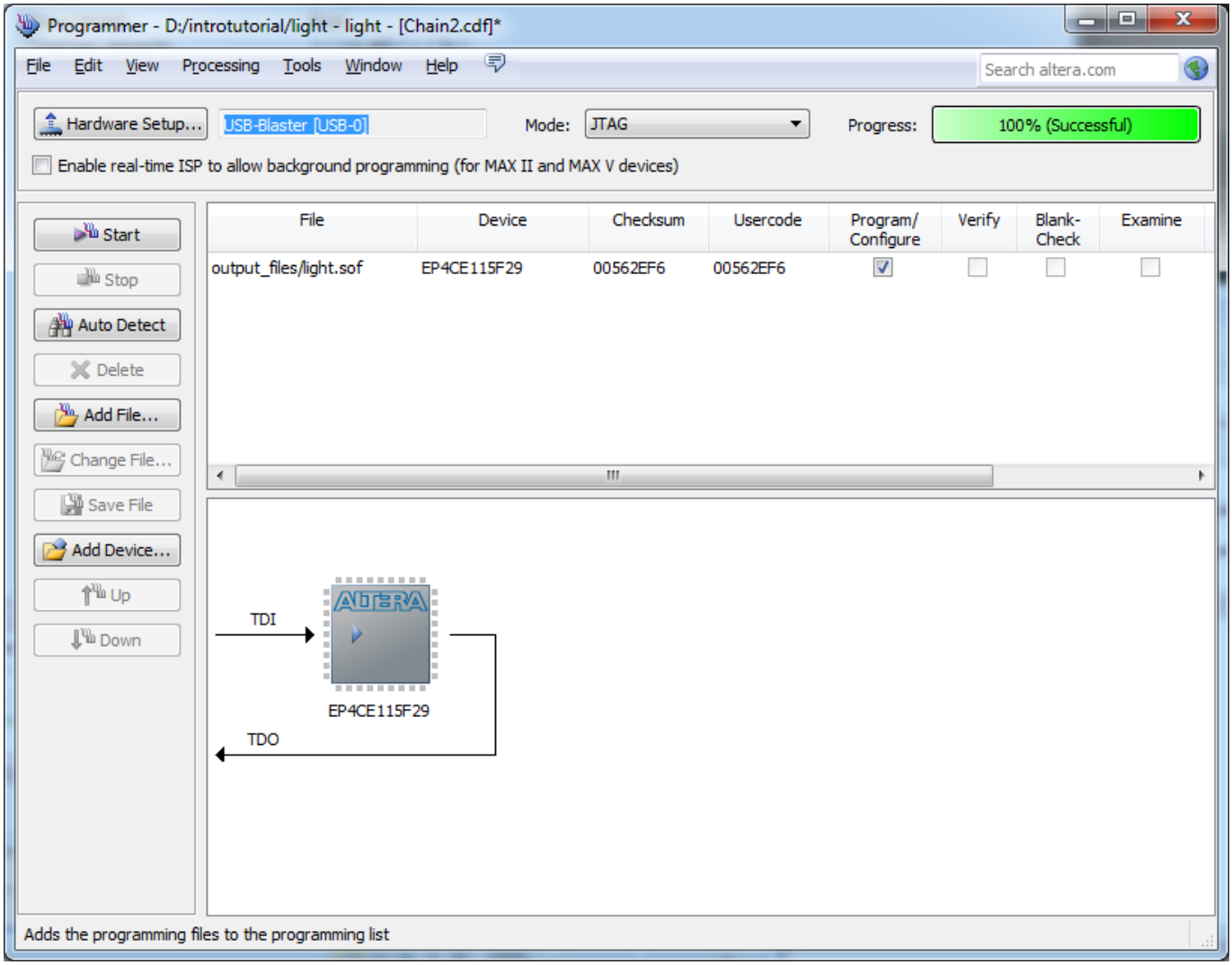


Figure 34. The updated Programmer window.

9. Testing the Designed Circuit

Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. Make sure that the RUN/PROG switch is at the RUN position. Try all four valuations of the input variables x1 and x2, by setting the corresponding states of the switches SW1 and SW0. Verify that the circuit implements the truth table in Figure 9. If you want to make changes in the designed circuit, first close the Programmer window. Then make the desired changes in the VHDL design file, compile the circuit, and program the board as explained above.

10. What to Hand In

Demonstrate your simulation and successful implementation on the DE2 board to the lab instructor for marking purposes. To get maximum marks, this lab should be completed before the end of your Lab session during week 2.

Reference

1. Quartus II Handbook available at the course web and Altera webpage