**Quartus Prime Introduction Using Verilog Design**

TCES 330

R. Gutmann

2/15/2018

This tutorial presents an introduction to the Quartus Prime 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 Prime 17 software. The design process is illustrated by giving step-by-step instructions for using the Quartus II and ModelSim software to implement and test a very simple circuit in an Altera FPGA device.

The Quartus II system includes full support for all the popular methods of entering a description of the desired circuit into a CAD system. This tutorial makes use of the Verilog design entry method, in which the user specifies the desired circuit in the System Verilog hardware description language. Two other versions of this tutorial are also available; one uses the VHDL hardware description language and the other is based on defining the desired circuit in the form of a schematic diagram. We will only be using (System) Verilog for hardware description and verification in this course.

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 DE2-115 Development and Education board connected to a computer that has Quartus II software installed.

The setups and procedures shown in this tutorial will be used for all the assignments, homework, labs, and exams in this course. So, keep this handy and refer to it as necessary.

**Contents:** Getting started

Starting a New Project Verilog Design Entry Compiling the Design Pin Assignment

Simulating the Designed Circuit

Programming and Configuring the FPGA Device

Testing the Designed Circuit

## 1 Getting Started

We will be using Quartus Prime 17.0 for this course. Download and install the free version this software on your laptop for easy access. This is the same version number that has been installed on the lab computers. Going back and forth between Quartus version numbers can cause problems, so we will all standardize on version 17.0 for this course. Note there are newer versions on the Altera web site, but don’t use those. The Altera tutorial *Getting Started with Intel’s DE-Series Boards* will help you through the process of installing Quartus Prime. Also the lab engineers can assist you.

Each logic circuit, or sub circuit, being designed with Quartus 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) and a couple of sub-directories. To begin a new logic circuit design, the first step is to create a directory (folder) to hold its files. To hold the design files for this tutorial, we will use a directory called HW0. The running example for this tutorial is a simple circuit for two-way light control, shown later.

**Note:** You will become hopelessly lost trying to figure out what file does what without making the following modification to your computer: **UNCHECK** “Hide extensions for known file types” in your Windows Explorer. Refer to Figure 1 and Figure 2, below for instructions. Microsoft hides extensions by default so as not to frighten your grandparents when they use a computer. You are expected to not have this problem.

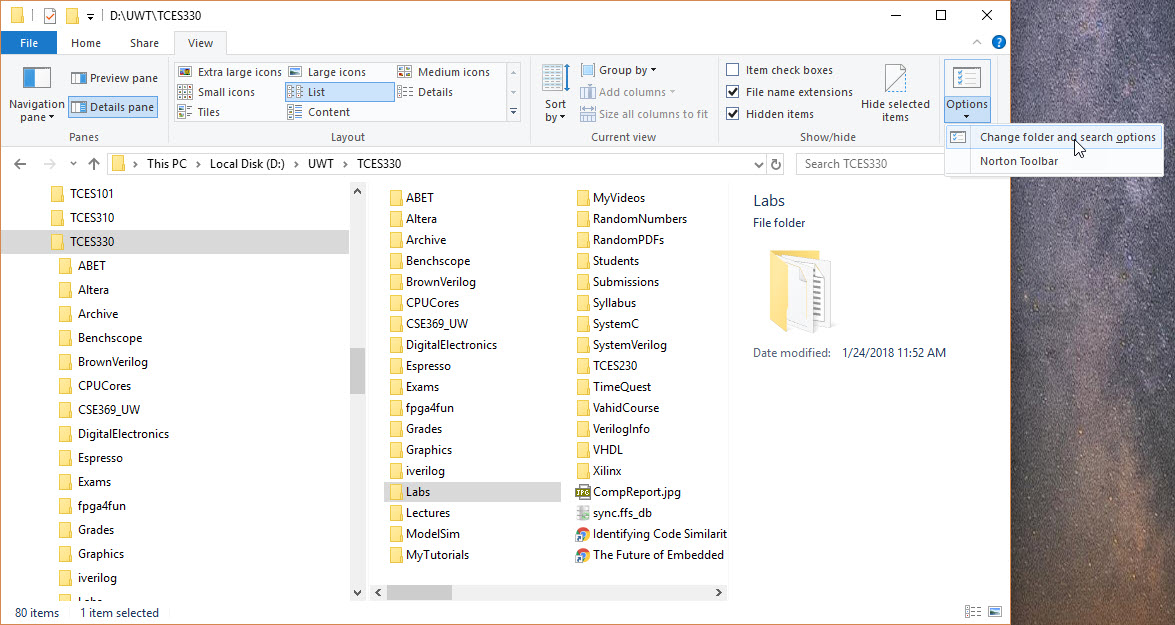


Figure . Navigate to the Options Menu as Shown

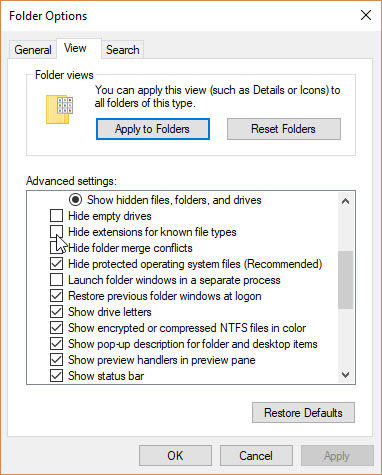


Figure . Find the 'Hide extensions' Checkbox and Uncheck

Start the Quartus Prime software. You should see a display similar to the one in Figure 3. This display consists of several windows that provide access to all the features of Quartus Prime software, which the user selects with the computer mouse. Most of the commands provided by Quartus Prime software can be accessed by using a set of menus that are located below the title bar. For example, in Figure 3 clicking the left mouse button on the menu named **File** opens the menu shown in Figure 4. Clicking the left mouse button on the entry **Exit** exits from Quartus Prime 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.

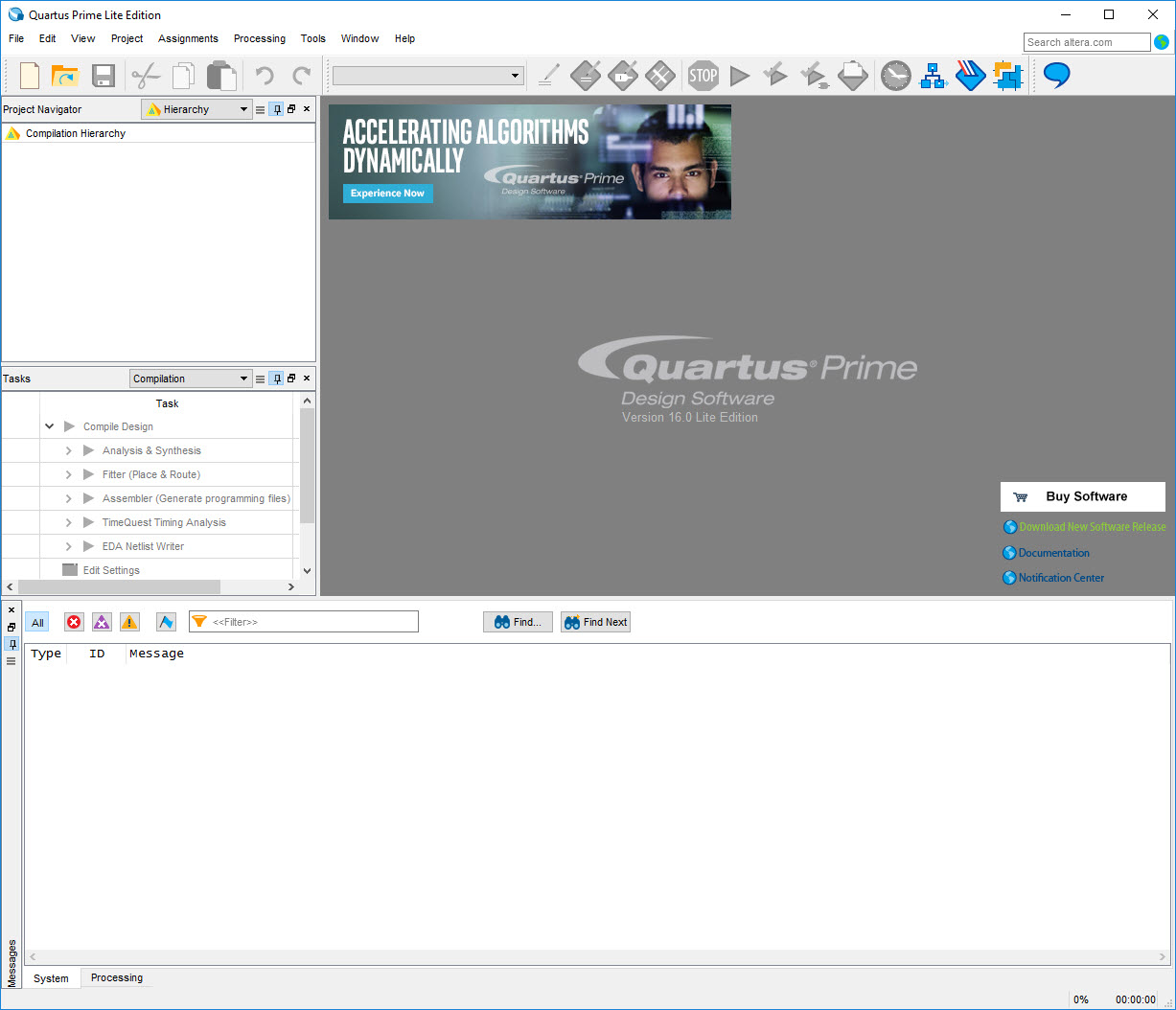


Figure . The main Quartus display.

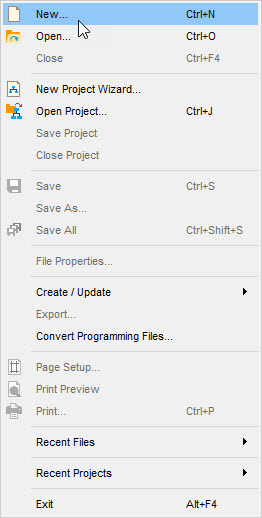


Figure . An example of the File menu.

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 Customizing.  
You may want to set up your version of Quartus to make it easier to use. Open **Tools > Options**. My **General** screen is shown in Figure 5. In particular, you should set the ‘Default file location’ to something close to where you will be saving your project files. This will save a lot of clicking around to get to your projects.

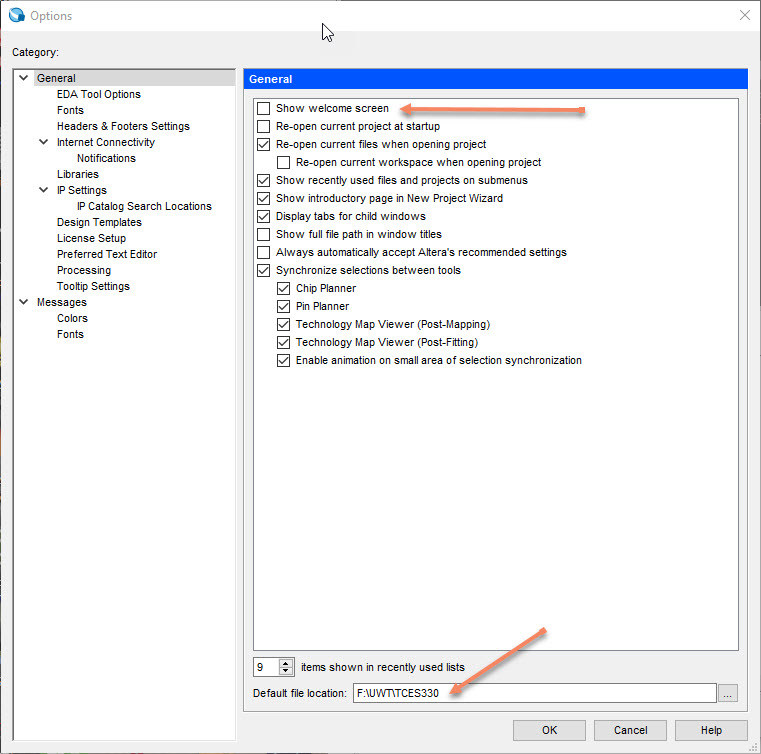


Figure . Quartus Options General

The EDA Tool Options dialog should show the path to where your ModelSim.exe file is located (note: ModelSim is installed when you install Quartus Prime). See Figure 6.This should already be filled in, but if not, enter (browse to) the correct path. Yours will be slightly different than the one show in the figure; it will, however, be in the …\modelsim\_ase\win32aloem folder.

I also use a custom browser to access Quartus Help. Refer to the Help section below (and Figure 7).

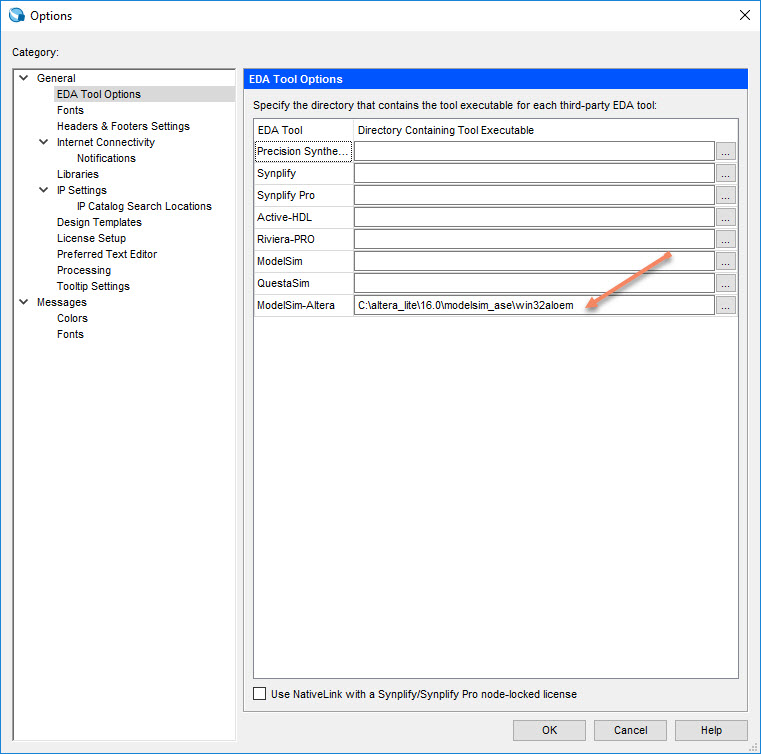


Figure . Quartus EDA Tool Options.

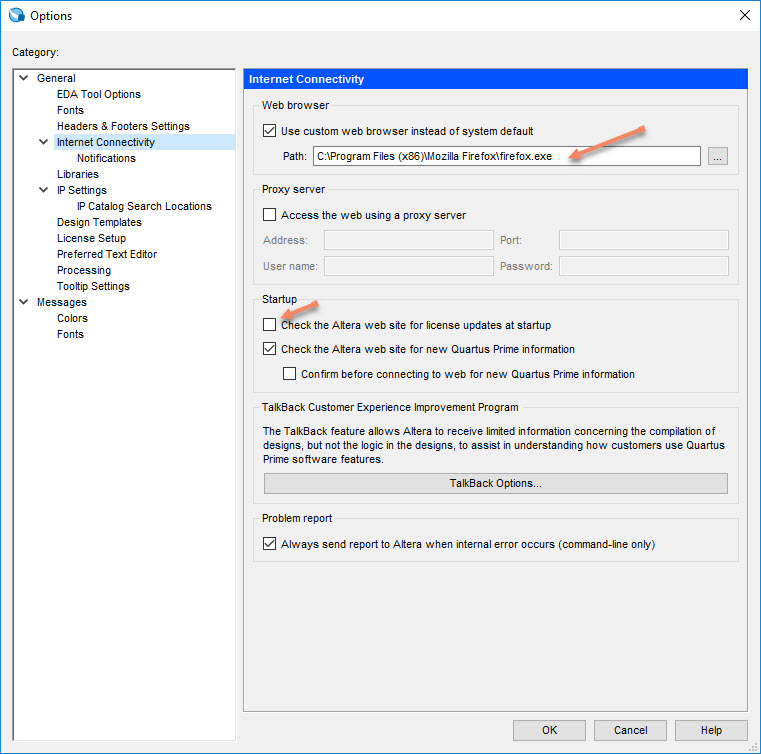


Figure . Quartus Options Internet Connectivity.

### 1.2 Quartus Prime 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 > Help Topics** gives an indication of what type of help is provided.

The user can quickly search through the Help topics by selecting **Help > Help Topics**, which includes a search box (upper right corner of the page) 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.

Note: In the past I have found that using Firefox for Quartus help worked best. Newer versions of Quartus seem to have resolved the issues with different browsers. However, you should tell Quartus which browser to use by going to **Tools > Options > Internet Connectivity** and entering the path to your favorite browser. Refer to Figure 7.

## 2 Starting a New Project

To start working on a new design we first must define a new design project. Quartus Prime 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 8, 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 9.

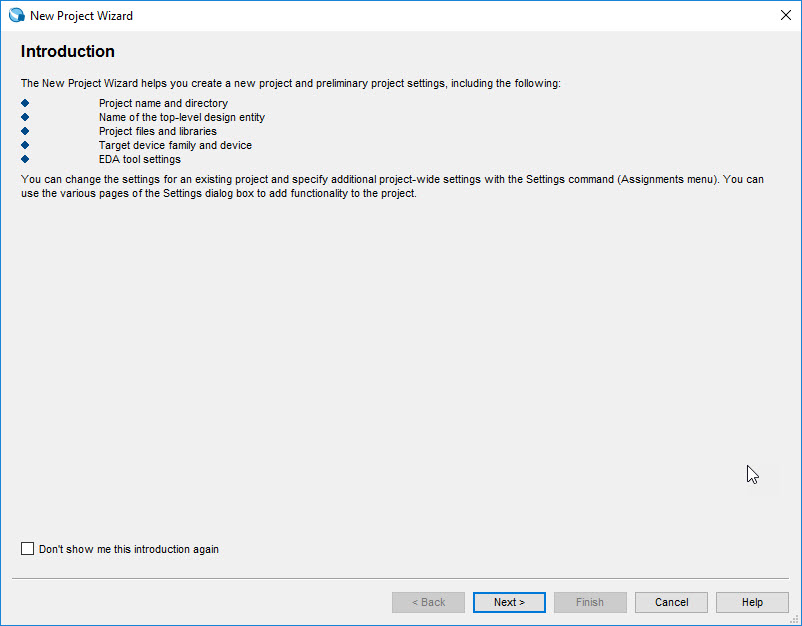


Figure . Tasks performed by the wizard.

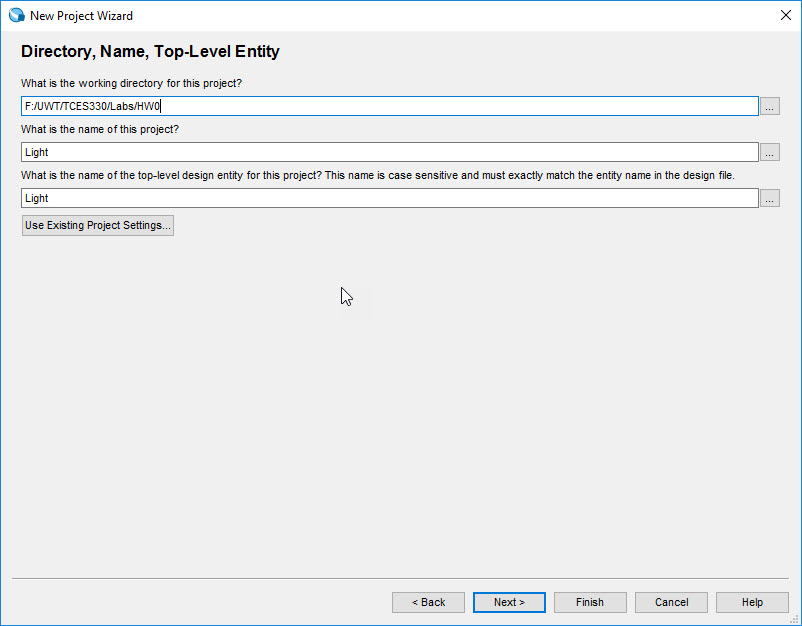


Figure . Creation of a new project.

1. Set the working directory to be HW0 as shown in Figure 9. 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 9. Press **Next**. If you have not yet created the directory HW0, the Quartus software displays the pop-up box in Figure 10 asking if it should create the desired directory. Click **Yes,** which leads to the window in Figure 11. Just keep ‘Empty project’ selected in this dialog.

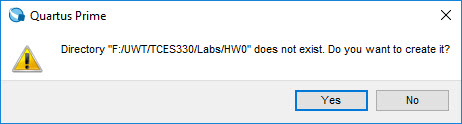


Figure . Quartus can create a new directory for the project.

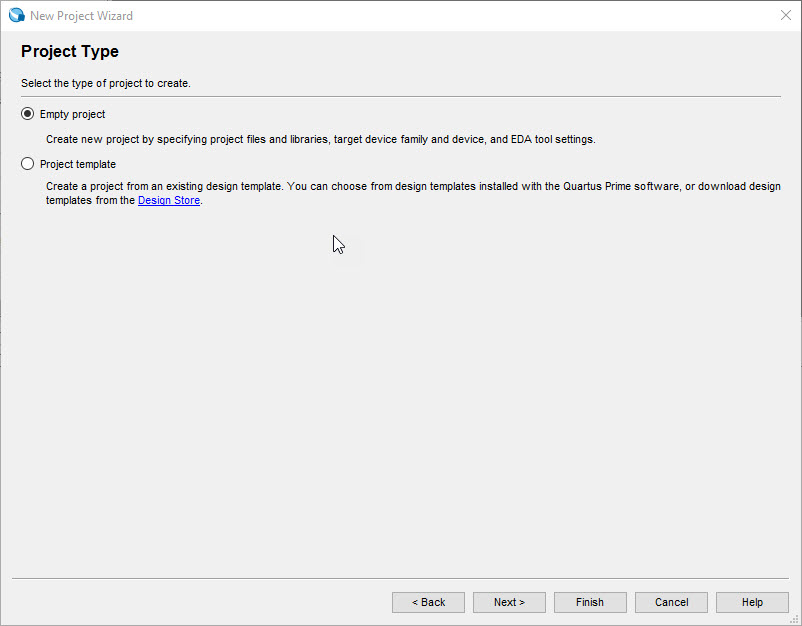


Figure . Quartus project type.

Figure 12 shows the next dialog, which lets you specify design (usually SystemVerilog) files. But since we don’t have any yet, just select ‘Next’ here, which leads us to Figure 13.

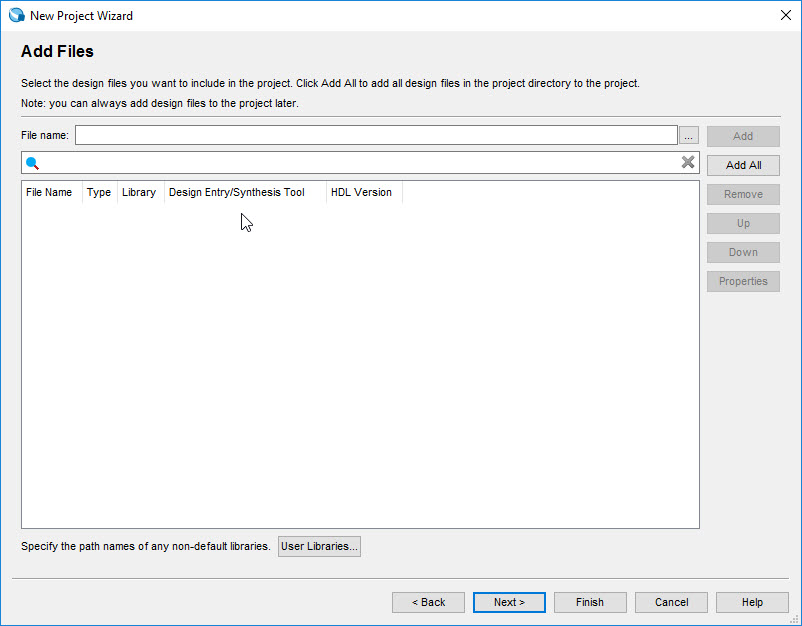


Figure . Quartus add design files.

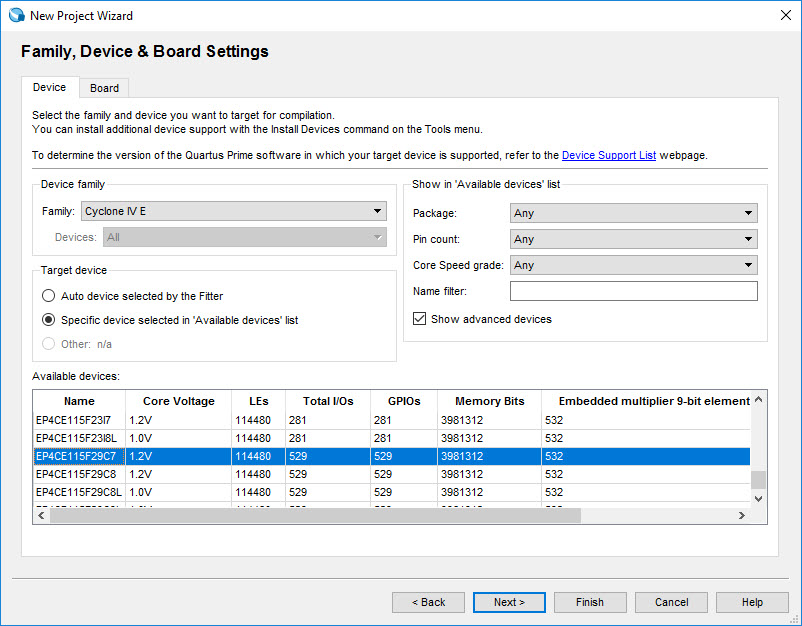


Figure . Quartus Device Selection

This dialog is fairly critical. For the DE2-115 boards you **must** select the device shown in Figure 13. Other devices will **not** work. Once you have made this selection, press ‘Next.’

Make sure the next dialog (Figure 14) shows ModelSim-Altera as the Simulation tool and SystemVerilog as the format HDL. Then press **Next**.

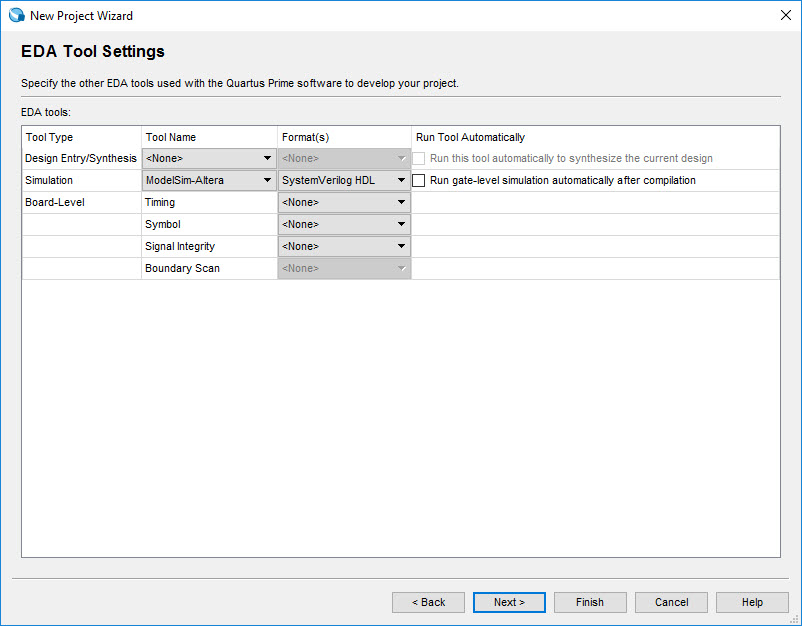


Figure . Quartus EDA Tool Settings

A summary of the chosen settings appears in the screen shown in Figure 15. Press

**Finish**, which returns to the main Quartus window, but with Light specified as the new project, in the display title bar, as indicated in Figure 16.

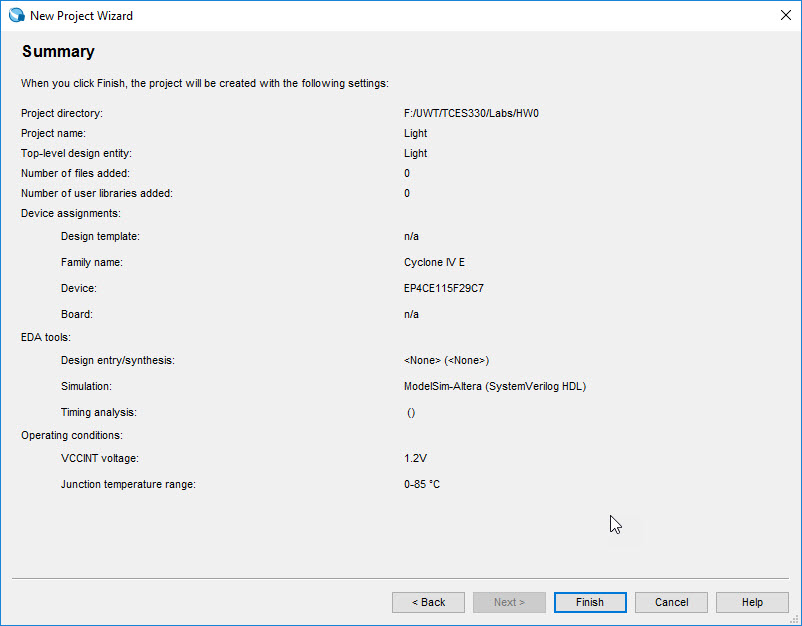


Figure . Summary of the project settings.

10

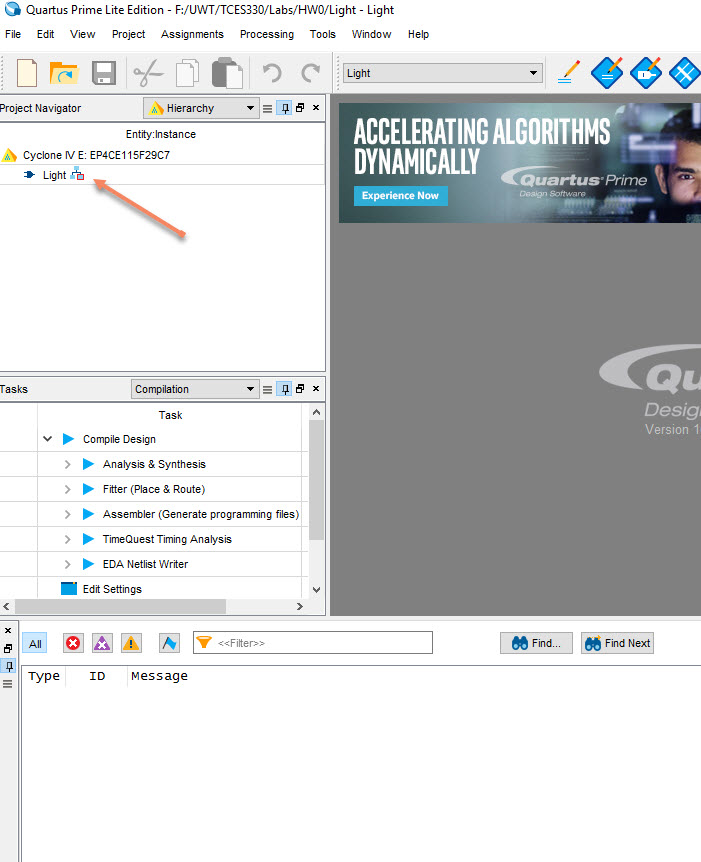


Figure . Quartus display for the created project.

## 3 Design Entry Using SystemVerilog Code

As a design example, we will use the two-way light controller circuit shown in Figure 17. The circuit can be used to control a single light from either of the two switches, X1and X2, 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 X1and X2, but we will specify it using the gates shown.

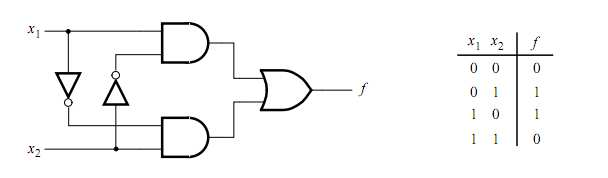


Figure 17. The Light controller circuit.

The required circuit is described by the SystemVerilog code in Figure 18. Note that the module is called Light to match the name given in Figure 16, which was specified when the project was created. This code can be typed into a file by using any text editor that stores **ASCII** files, or by using the Quartus Prime 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 Verilog module. The file name must include the extension sv, which indicates a SystemVerilog file. So, we will use the name Light.sv.

module Light(F, X1, X2 );

input X1, X2;

output F;

assign F = (X1 & ~X2) | (~X1 & X2);

endmodule

Figure 18. SystemVerilog code for the light controller circuit.

### 3.1 Using the Quartus Prime Text Editor

This section shows how to use the Quartus Prime Text Editor. You can skip this section if you prefer to use some other text editor to create the Verilog source code file, which we will name Light.sv. Select **File > New** to get the window in Figure 19, choose SystemVerilog HDL 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** to open the pop-up box depicted in Figure 20. In the box labeled **Save as type** choose SystemVerilog HDL 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 HW0 and leads to the Text Editor window shown in Figure 21.

Maximize the Text Editor window and enter the Verilog code in Figure 18 into it. Save the file by typing **File > Save**, or by typing the shortcut **Ctrl-s**. Most of the commands available in the Text Editor are self-explanatory. Text is entered at the insertion point, which is indicated by a thin vertical line. The insertion point can be moved either by using the keyboard arrow keys or by using the mouse. Two features of the Text Editor are especially convenient for typing Verilog code. First, the editor can display different types of Verilog statements in different colors, which is the default choice. Second, the editor can automatically indent the text on a new line so that it matches the previous line. Such options can be controlled by the settings in **Tools > Options > Text Editor**.

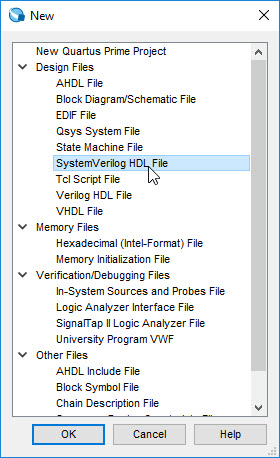


Figure . Choose new SystemVerilog File.

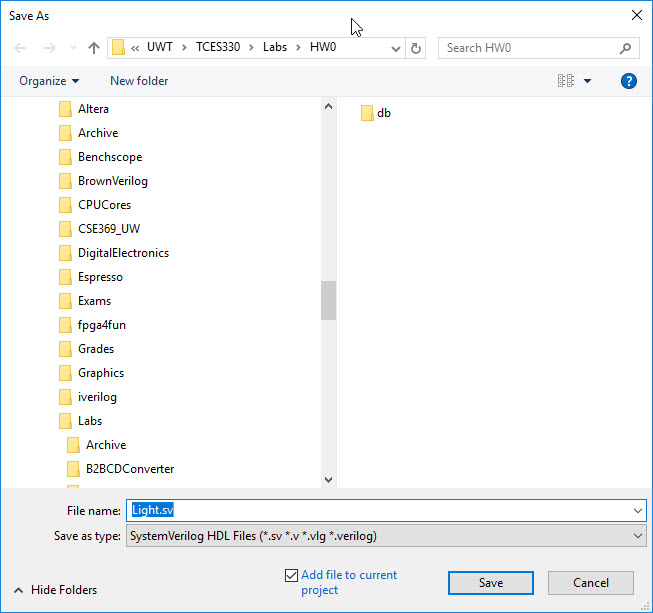


Figure . Name the file.

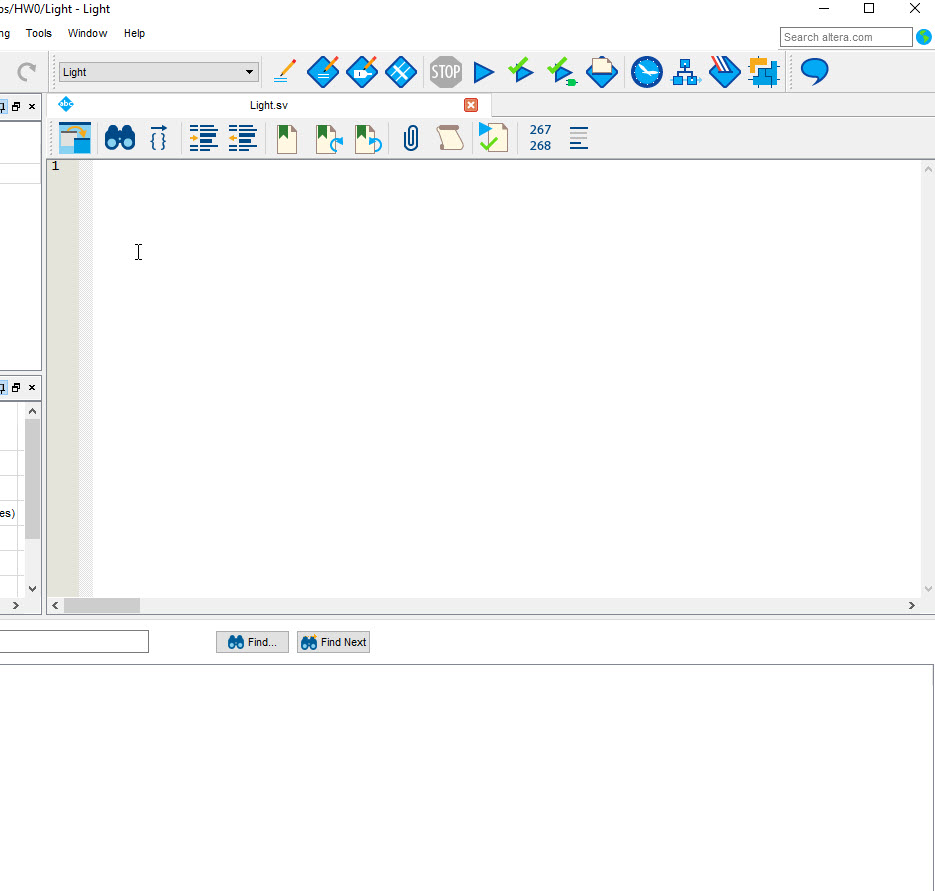


Figure . Quartus Text Editor Window.

### 3.1.1 Using Verilog Templates

The syntax of Verilog code is sometimes difficult for a designer to remember. To help with this issue, the Text Editor provides a collection of SystemVerilog templates (as well as Verilog and VHDL templates). The templates provide examples of various types of SystemVerilog statements, such as a **module** declaration, an **always** block, and assignment statements. It is worthwhile to browse through the templates by selecting **Edit > Insert Template > SystemVerilog** to become familiar with this resource. On the other hand, it’s to your advantage to **not** rely too much on this resource; you should know the basic Verilog constructs.

### 3.2 Adding Design Files to a Project

As we indicated when discussing Figure 12, you can tell 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 22. 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 you used the Quartus Text Editor to create the file and checked the box labeled **Add file to current project**, as described in Section 3.1, then the Light.sv file is already a part of the project and will be listed in the window in Figure 22. Otherwise, the file must be added to the project. So, if you did not use the Quartus Text Editor, then place a copy of the file Light.sv, which you created using some other text editor, into the directory HW0. To add this file to the project, click on the **File name**: button in Figure 22 to get the pop-up window in Figure 23. Select the Light.sv file and click **Open**. The selected file is now indicated in the Files window of Figure 22. Click **Add** to include the Light.sv file in the project (this step is easy to miss).

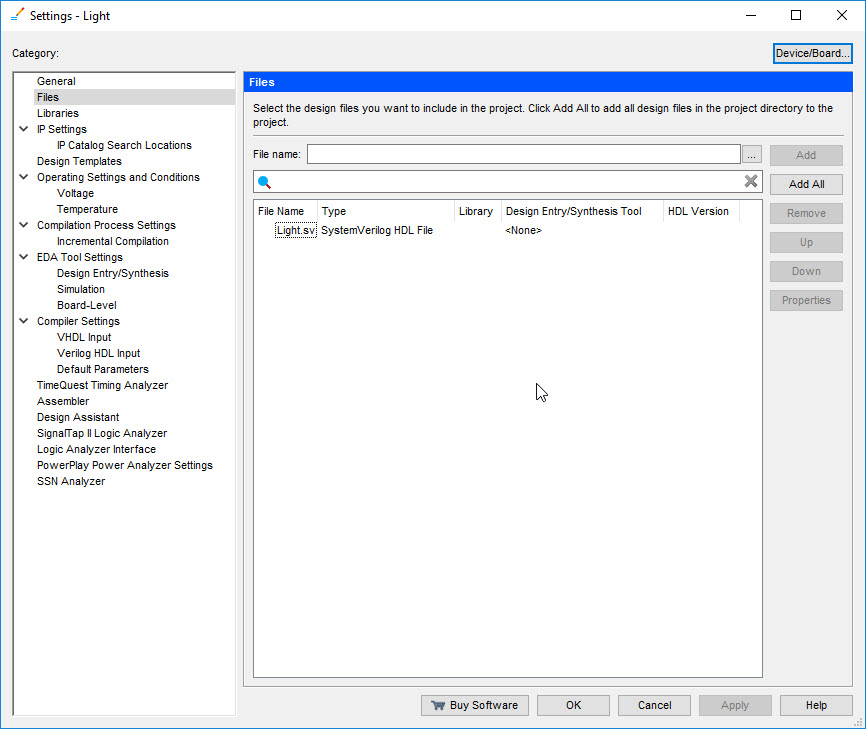


Figure . Quartus Files Settings.

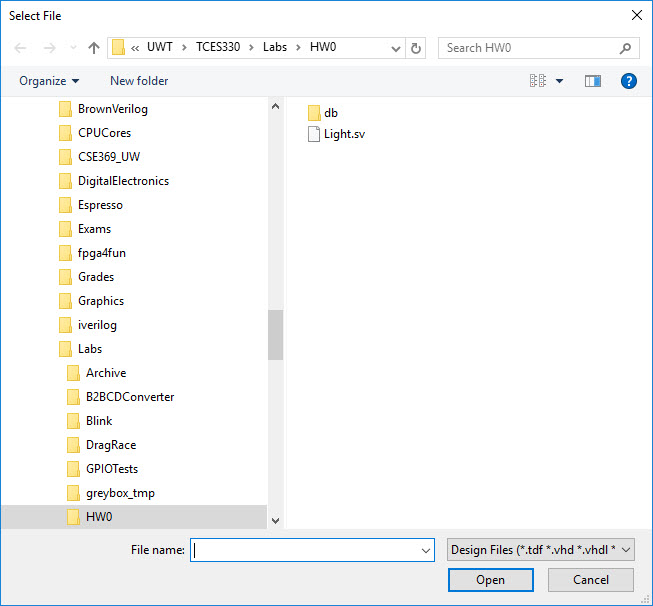


Figure . Select file(s).

We are not quite done with Light.sv yet. We will develop the habit of testing each module we write. Although you won’t find this ruled followed everywhere (especially in academic settings!), it is the sort of thing expected of professional designers. To that end, add the following ‘testbench’ code to the end of your Light.sv file.

// Testbench

module Light\_testbench();

reg X1, X2;

wire F;

Light DUT( .X1(X1), .X2(X2), .F(F) ); // strange right?

initial begin

X1 = 0; X2 = 0; #10;

X1 = 0; X2 = 1; #10;

X1 = 1; X2 = 1; #10;

X1 = 1; X2 = 0; #10;

end

endmodule

**Figure 24. Testbench code for module Light.**

In class we will discuss in detail what the testbench actually does. But most of the work happens between the initial begin and end statements. These lines, as you can see, set X1 and X2 to all possible values. The #10 just means pause for 10 units of time so we can take a look at what’s happening. This should be a pretty thorough test of our circuit.

## Testing the Circuit

We will do two kinds of testing now. We will check the syntax to make sure we have written valid SystemVerilog code. Then we will test the functionality of the code to see if it does what we expect it to. At this stage we will ignore the subtitles introduced by gate delays. We will pick timing checks later in the quarter after we get into sequential circuits.

### 4.1 Syntax Check.

We will check the syntax by running the circuit through an Analysis and Synthesis step. Refer to Figure 25. The easiest way to do this is to pick the toolbar icon shown in the figure. Any syntax errors will show up in red as you can see in Figure 26 (where a semi colon was left off the last line for demonstration purposes).

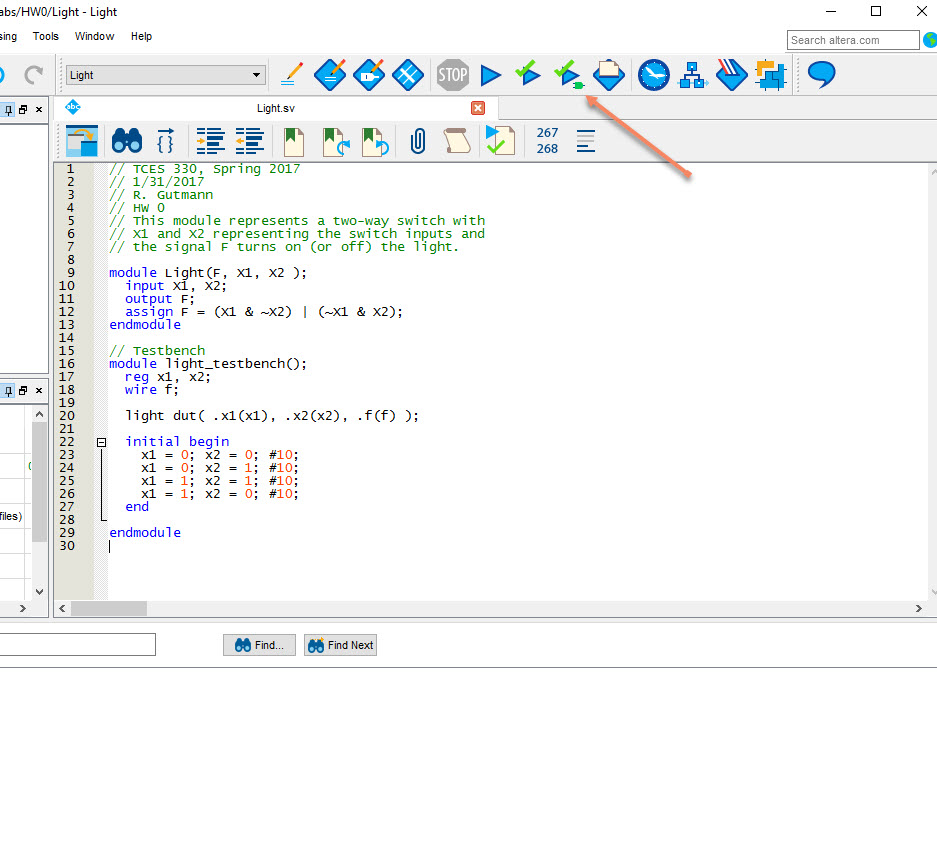


Figure . Quartus start Analysis and Synthesis.

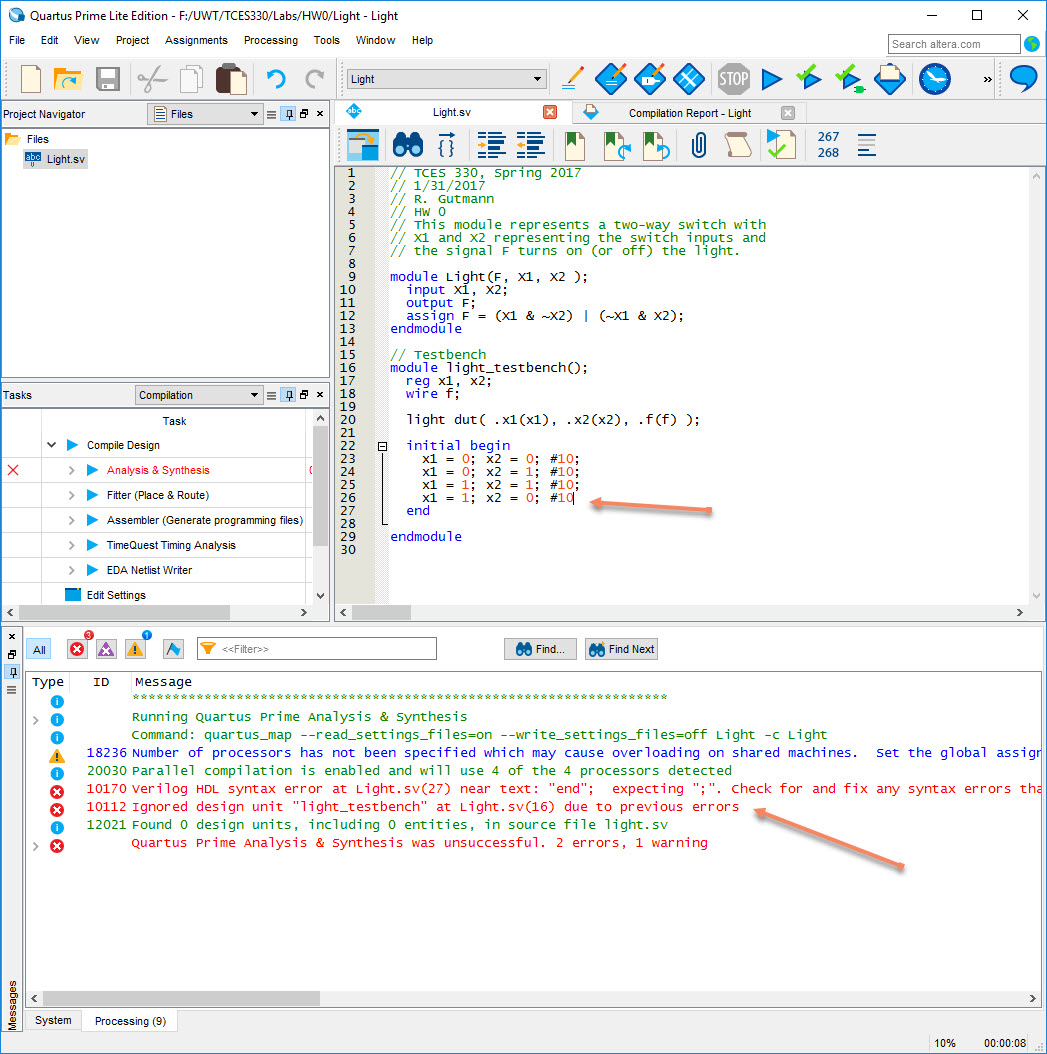


Figure . Example syntax error.

Fix any syntax errors and we will move on to functionality testing.

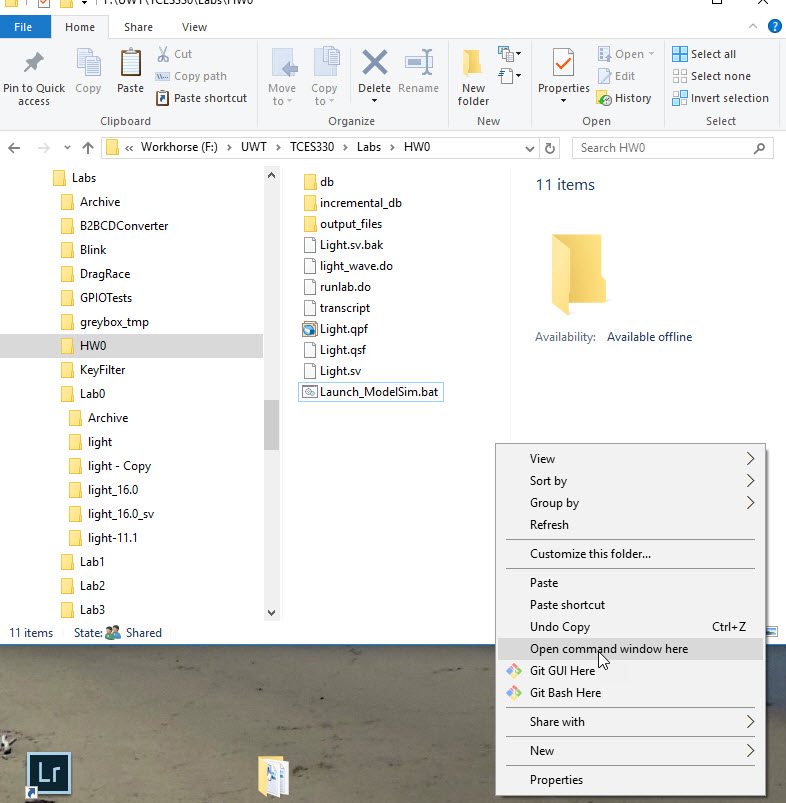
### 4.2 Functionality Testing

There are entire books written on and careers made in testing (System)Verilog code. We will keep it simple here, though. On Canvas I have provided three files to support this test as follows. Put all three of these files in your HW0 folder.

* Launch\_ModelSim.bat: This file just starts the ModelSim program in the current (HW0) folder. You may have to edit this file to point to wherever your modelsim.exe file is located. It will be something similar to what’s in the file I gave you, though.
* runlab.do: This is a ModelSim script file that compiles the design in ModelSim, sets up some windows, and starts the simulation.
* Light\_wave.do: This is another ModelSim script that sets up the graphical output for our simulation of Light.

We will talk much more about what’s in these scripts and how to modify them for the modules we will write later on.

For now, change to your HW0 folder if you are not already in it. Hold down the Shift key and right-click in the HW0 window. In the context menu that pops up you will see either **Open Command Window here** or **Open PowerShell here**. Select whichever one you see. Refer to **Figure 27**. In the whichever command window that opens, enter ./Launch\_Modelsim.bat and press return (or just enter La then press <tab>) (Figure 28 or Figure 29).



**Figure 27. Open Command Window in HW0.**

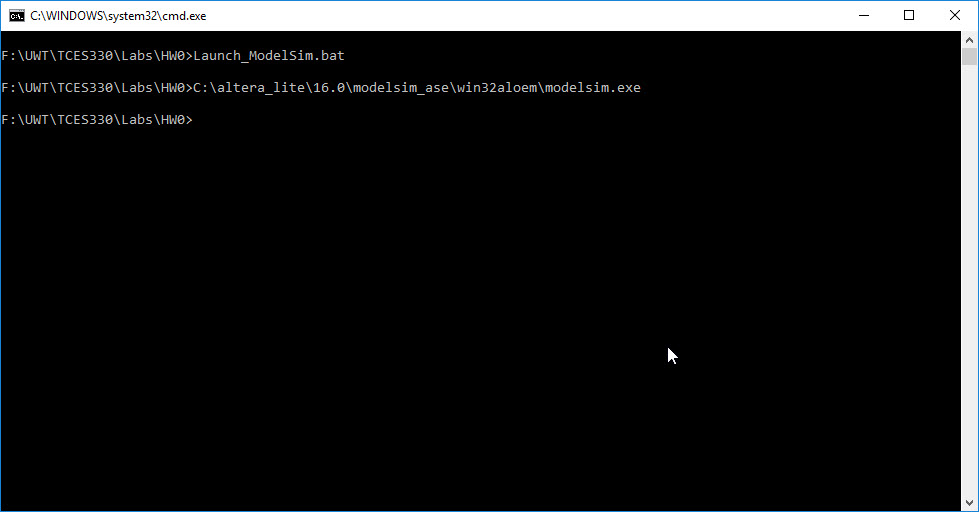


Figure . The Command Window.

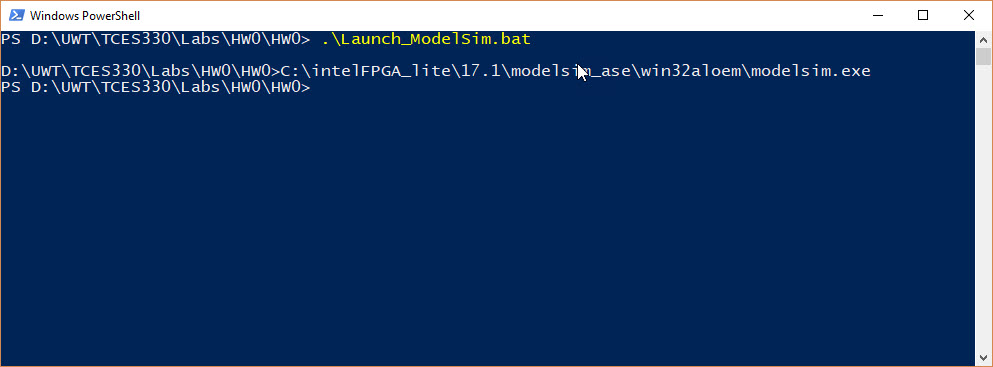


Figure . The PowerShell Window

If you see something in the command window about not knowing what modelsim.exe is all about, then you need to edit Launch\_Modelsim.bat to point to it. Get help if you don’t understand how to do this.

If everything works out, you should see the ModelSim window as shown in Figure 30.



Figure . ModelSim at start up.

In the Transcript window type in do runlab.do (or just do ru then press <tab> key) and press **Return**. See Figure 31. This should cause the simulation to run.

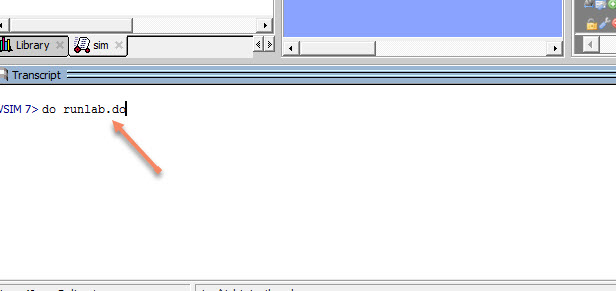


Figure . do runlab.do

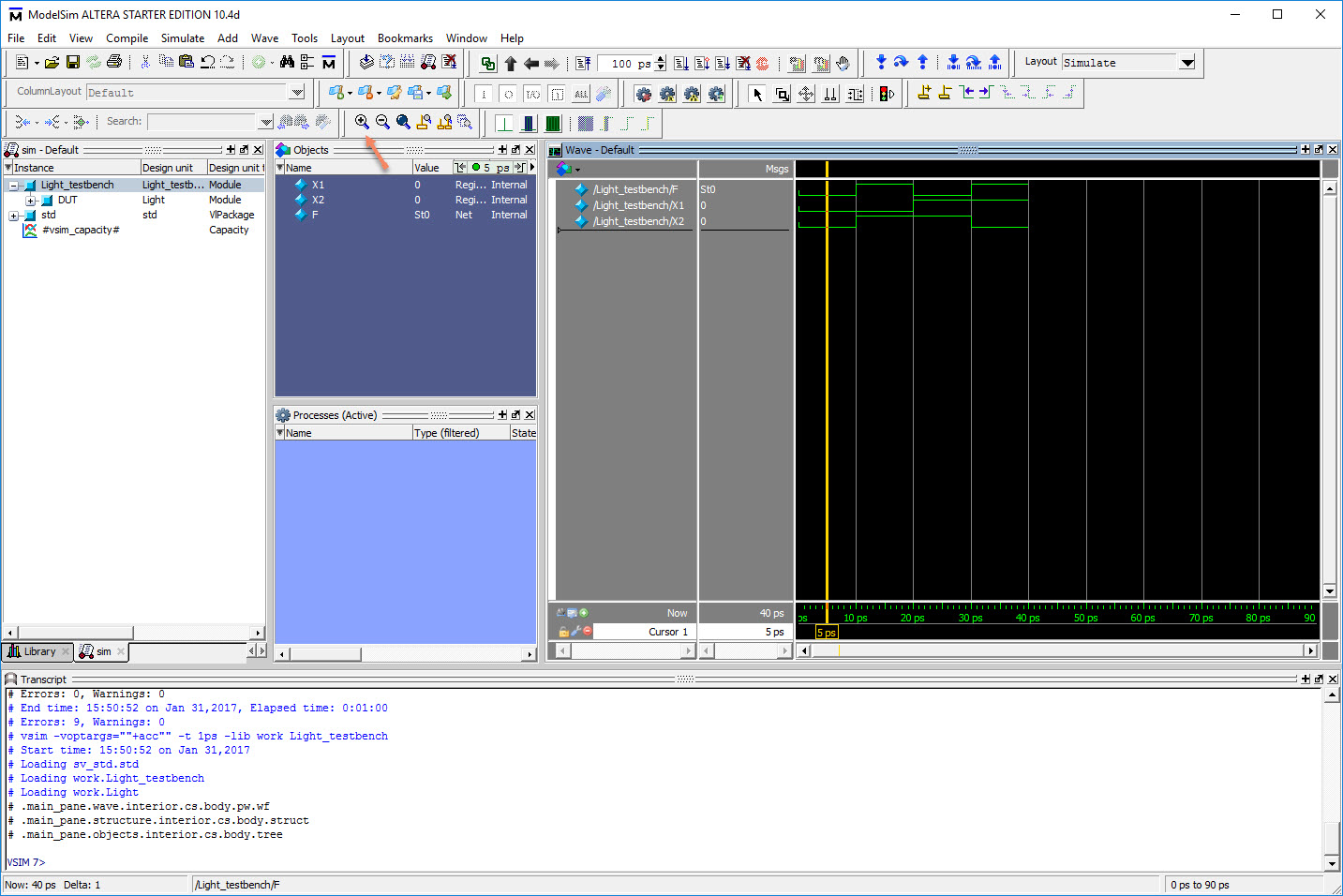


Figure . Simulation results.

This should cause the simulation to run. If there are red error messages in the Transcript window, then some error occurred. See if you can figure out what it was; if you can’t ask for help. Once you’ve fixed the errors, run do runlab.do again. When everything works, you should see something like Figure 32.

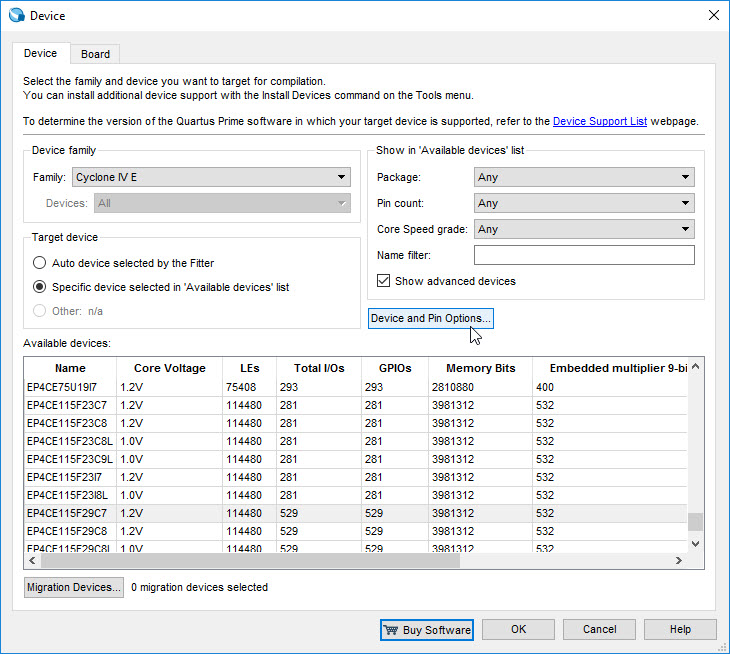
Note: You may have to click in the black graph window and press the ‘+’ icon (see the red arrow in Figure 32) a couple of times to expand the display. The top line of the graph is F, the output, and the two inputs are X1 and X2. You can easily verify that this circuit is working like an exclusive-OR, just what we were expecting. So, our circuit appears to be functional. You can close out ModelSim at this point.

### 4.3 Testing on the DE2-115 Board

Now let’s upload our design to the board. Back in Quartus we need to take care of a couple of things, and then do a full compile of our circuit.

First, let’s take care of a little hygiene. We will need to do for every design we put on the DE2. Select **Assignments > Device**, then select **Device and Pin Options**. See **Figure 33**. Then select **Unused Pins** and **Reserve all unused pins** and **As input tri-stated**. See Figure 34. Then **OK** all the way back out to the main Quartus window.

Taking care of these unused pins will turn off annoying unused outputs on the DE2 board. You will be required to do this for every design you turn in this quarter.



**Figure 33. Select Device and Pin Options.**

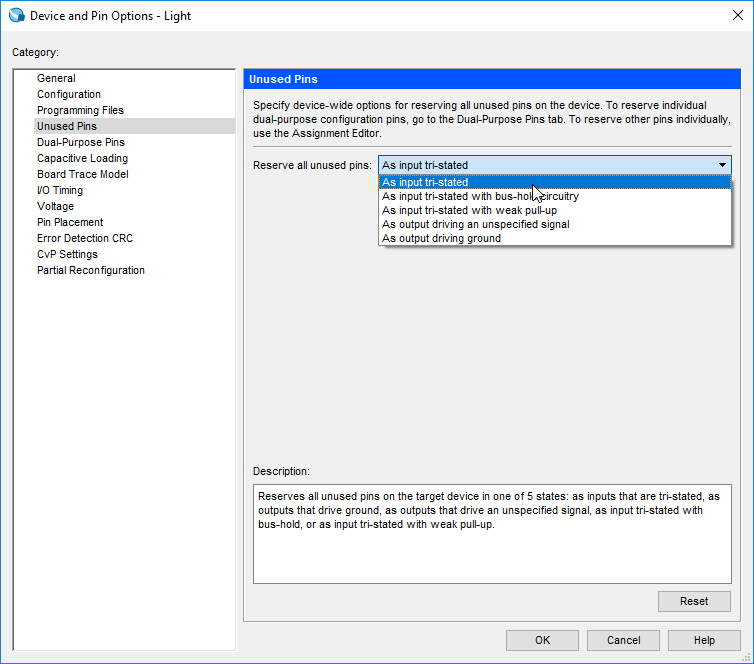


Figure 34. Select Input tri-stated.

So much for unused pins. What about pins we actually want to use?

### 4.3.1 Assigning Pins

At this point in our design, Quartus does not know where we expect the inputs X1 and X2 to come from, nor does it know how to display the output F. We need to tell it how to do these things.

We will use two toggle switches, labeled **SW0** and **SW1**, to provide the external inputs, X1 and X2, to our example circuit. Refer to Figure 35.

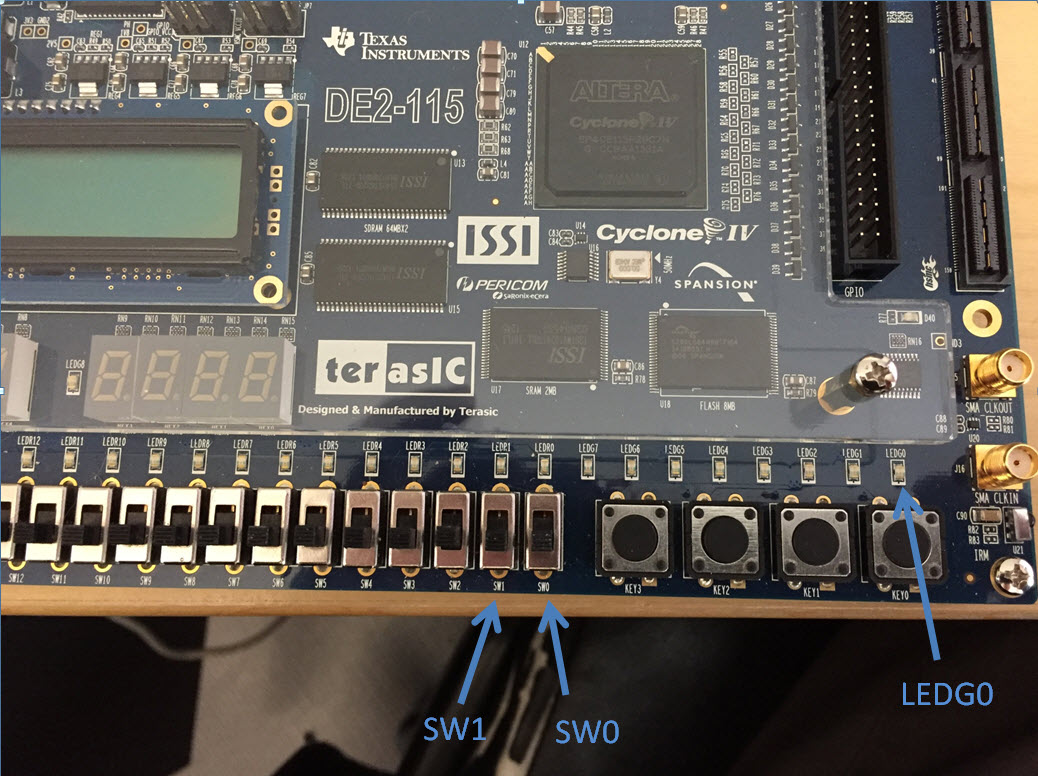


Figure . DE2-115 switches and LED.

These switches are connected to the FPGA pins called **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. Select **Assignments > Pin Planner** to reach the window in Figure 36. In the row labeled F (our output) double click in the Location column. The drop-down menu in Figure 37 appears.

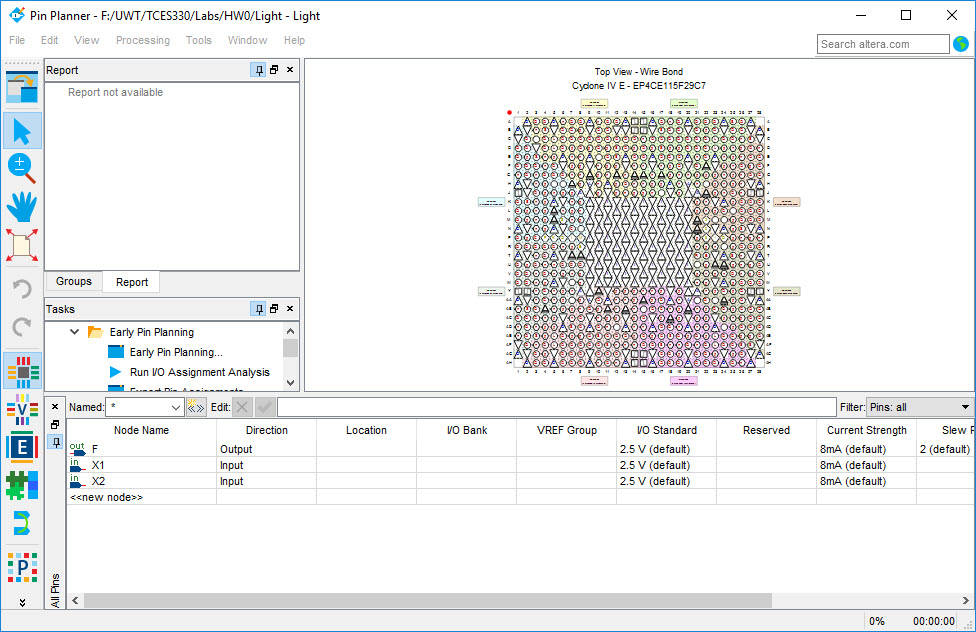


Figure . Quartus Pin Planner

Scroll down and select **PIN\_E21** (or type in PIN\_E21). Use the same procedure to assign input X1 to pin AB28 and input X2 to pin **AC28,** which results in the image in Figure 38. 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**.

As we shall see soon enough, there are easier ways to make pin assignments if we use a set of standard names for the lights and switches on the DE2-115 board. For now, it’s good enough for us to use the pin planner.

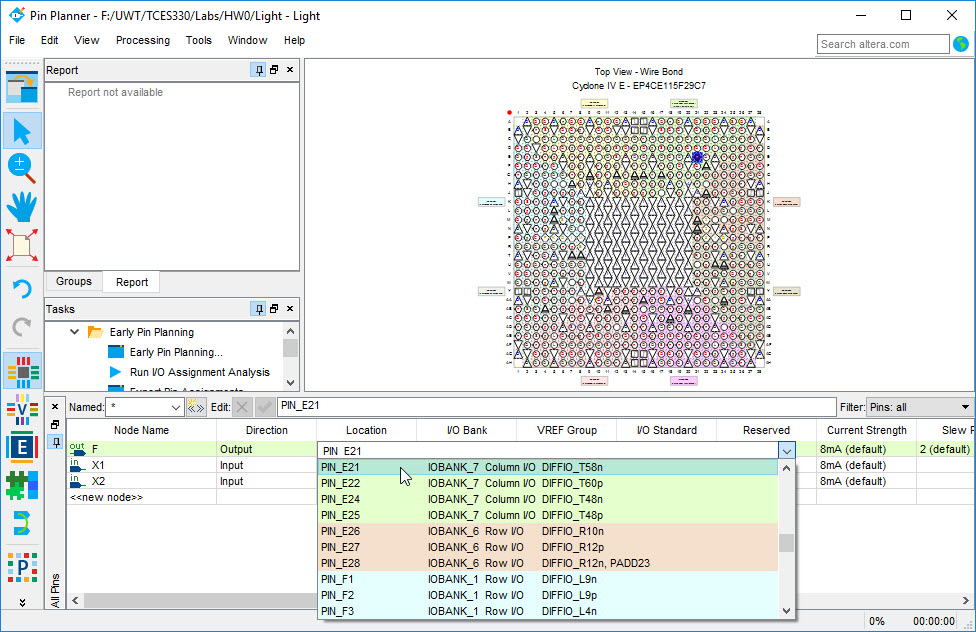


Figure 37. Drop-down menu displays pins.

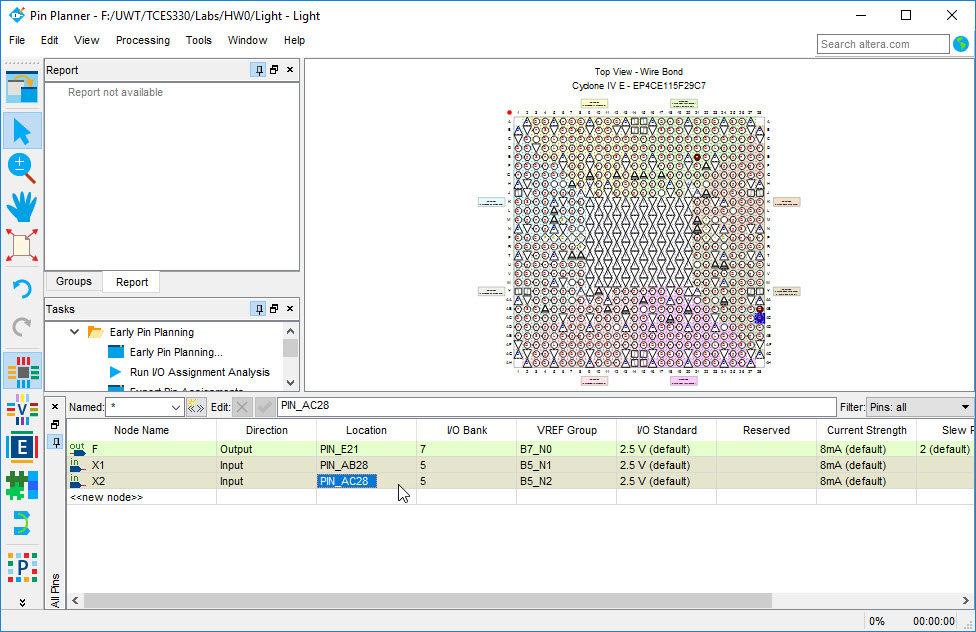


Figure 38. The complete assignment.

### 4.3.2 Compiling the Designed Circuit

The Verilog code in the file Light.sv is processed by several Quartus tools that analyze the code, 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 blue triangle (or the ‘play’ button on your DVD player). As the compilation moves through various stages, its progress is reported in a window on the left side of the Quartus display. Completion of the compile process is indicated by ‘100%’ in the lower right corner. Refer to the arrows in Figure 39. Compilation errors will be shown as red text messages in the message window at the bottom; warnings will be shown in blue.



Figure . Quartus compiling.

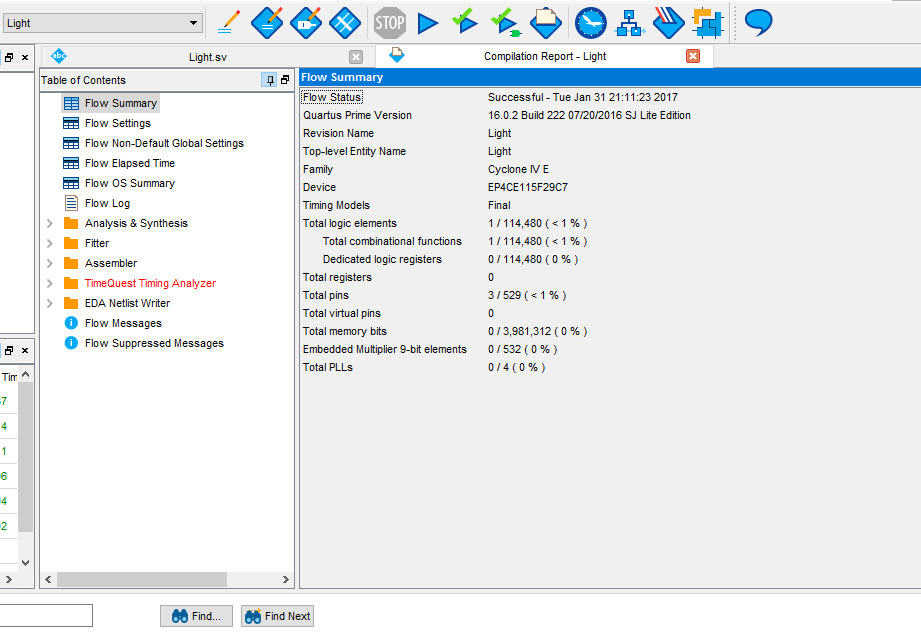


Figure . Quartus Compilation Report.

When the compilation is finished, a compilation report is produced. A window showing this report is opened automatically, as seen in Figure 40. 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 40 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.

### 4.3.3 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. We will only concern ourselves with the JTAG mode here. 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 DE2-115 Board or one of the lab engineers 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. Your DE2 board flash memory should contain the start-up circuit that flashes all the lights and hex displays. If it does not, check with the lab engineers about getting this installed – it’s a good way to make sure all the lights and displays are working and may save you considerable debugging time if some component should fail.

The RUN/PROG switch on the DE2-115 board should be in the RUN position, which selects the JTAG mode.

#### 4.3.3.1 JTAG Programming

The programming and configuration task is performed as follows. Flip the RUN/PROG switch into the RUN position. Select Tools > Programmer (or click on the Programmer icon as shown in Figure 41) to reach the window in Figure 42. 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, the box pointed to by the red arrow in Figure 42 may say ‘No Hardware.’ In this case, press the ‘Hardware Setup...’ button and select the USB-Blaster in the window that pops up, as shown in Figure 43. If this drop-down does not show the USB-Blaster as a choice, you need to install or update the driver for this device. Once again, check with the lab engineers for information about how to accomplish this. When all of this is done, your Programmer Window should look like Figure 42.

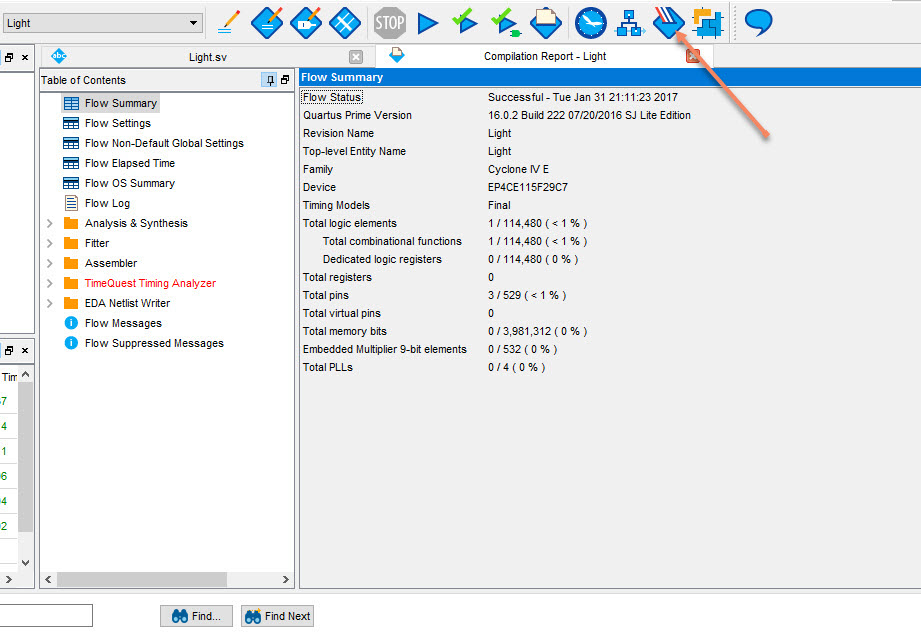


Figure . Quartus Programmer Icon.

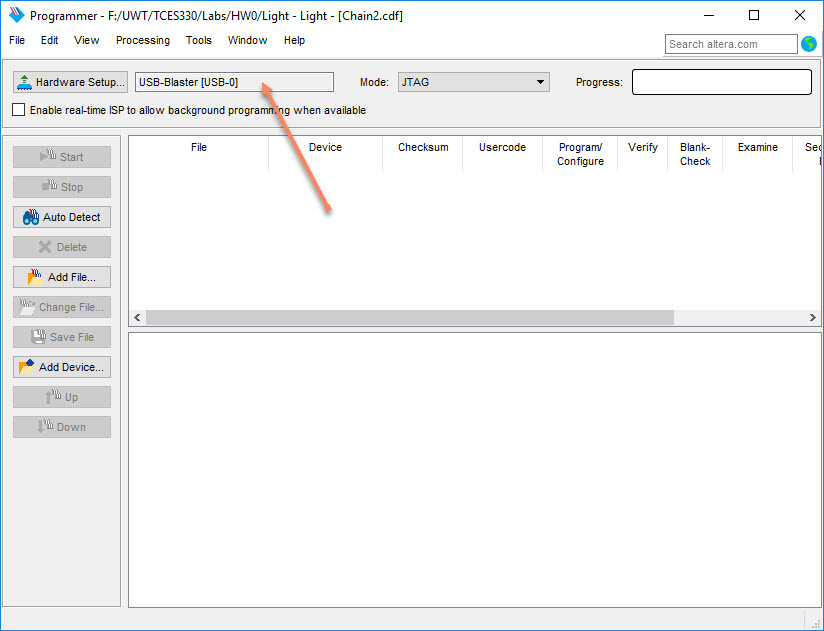


Figure . The Programmer window.

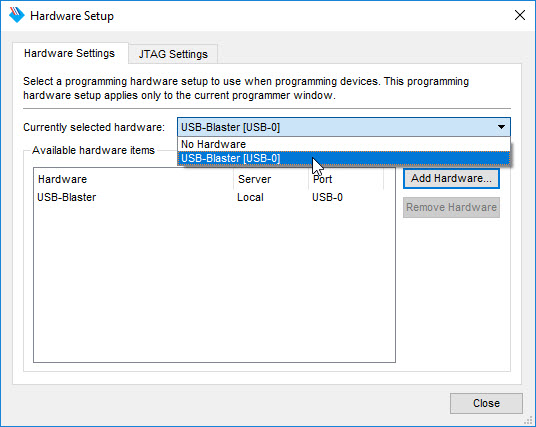


Figure . The Hardware Setup window.

Now we need to add the file we want uploaded to the board to the Programmer Window. As shown in Figure 44, click on the **Add File** button. A File Selection dialog will pop up. Double click on the ‘output\_files’ folder and select Light.sof. 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 EP4CE115F29C7, which is the FPGA device used on the DE2-115 board.

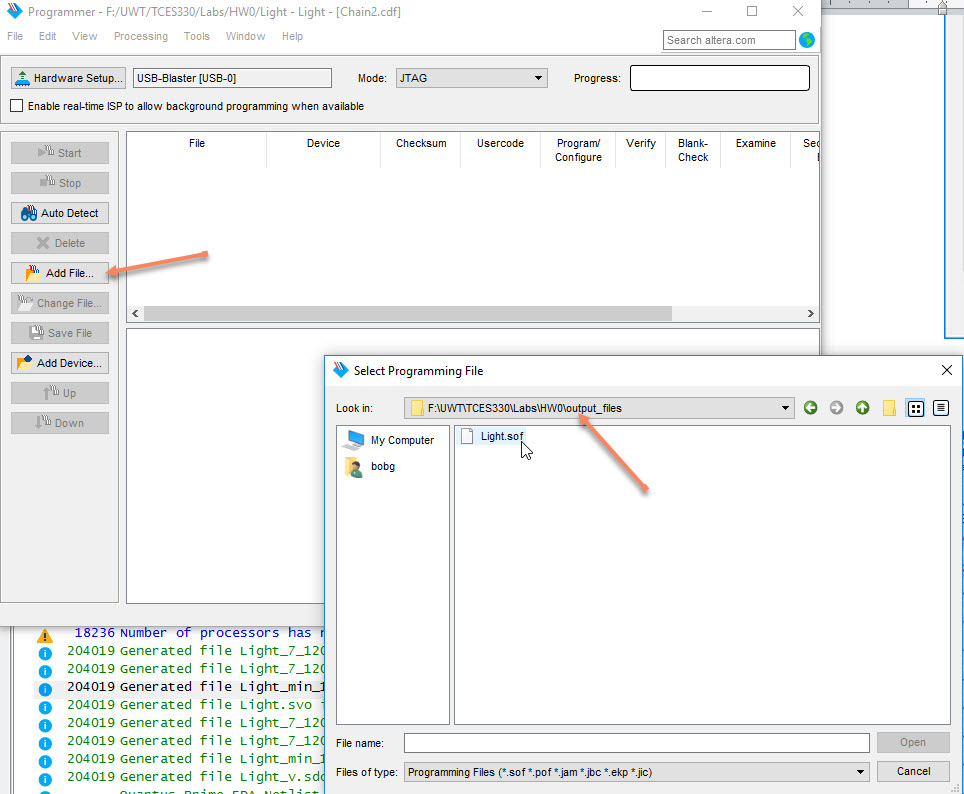


Figure . Selecting the sof file.

Now, press **Start** in the window in Figure 45. The ‘Progress’ box will turn green and will say ‘100% (Successful)’ to indicate success (see Figure 46). If it says ‘Failed’ then try pressing **Start** again. For some reason it’s not unusual to have to press Start a couple of times. If you continue to have trouble with it, make sure the USB-Blaster cable is connected to the correct USB port on the board and that the board is powered on.

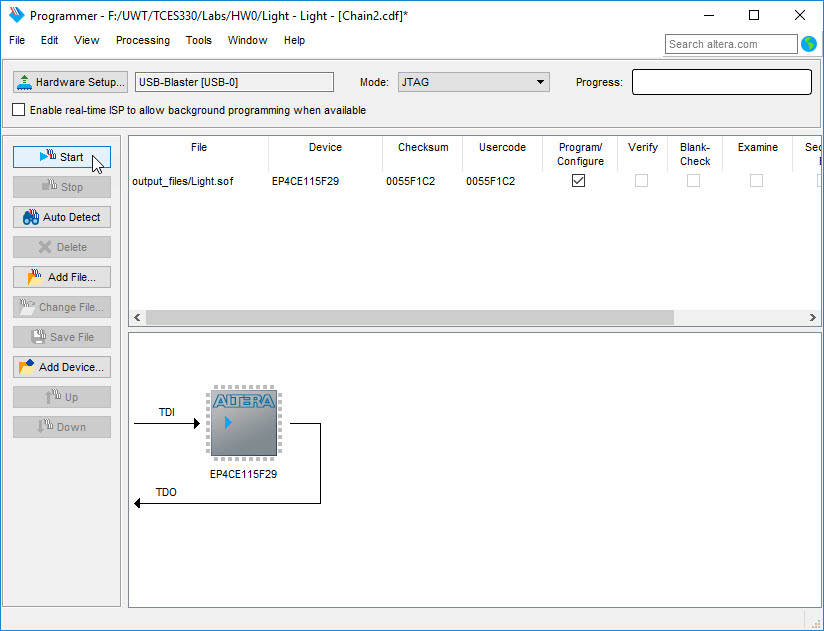


Figure . The updated Programmer Window.

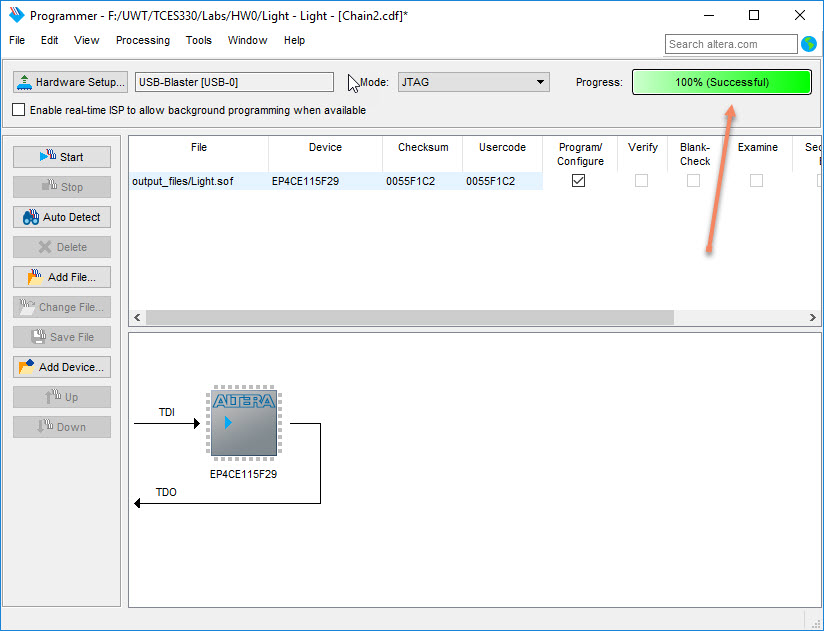


Figure . Programmer success.

### 4.3.4 Testing the Designed Circuit on the DE2

Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. Make sure the **RUN**/**PROG** switch is in the **RUN** position. Try all four valuations of the input variables X1 and X2 (just like we did in ModelSim), by setting the corresponding states of the switches **SW1** and **SW0**. Verify that the circuit implements the truth table in Figure 17.

If you want to make changes in the designed circuit, first close the Programmer window. Then make the desired changes to your Verilog codefile, compile the circuit, and program the board as explained above.

### Appendix A. Installing the DE2 Board Drivers

After a clean installation of Quartus, you will probably need to also install the DE2 board drivers. If you follow the directions of Section 4.3.3.1 and find there is no hardware listed, even in the dialog shown in Figure 43, you need to install the drivers. This is normal.

In the Windows 10 Search, type in ‘Device Manager’ as shown in Figure 47 and press Return. In the Device Manager you should see an entry with an exclamation in a triangle as shown in Figure 48. Right-click on this entry and pick ‘Update Driver’ as shown in Figure 49.

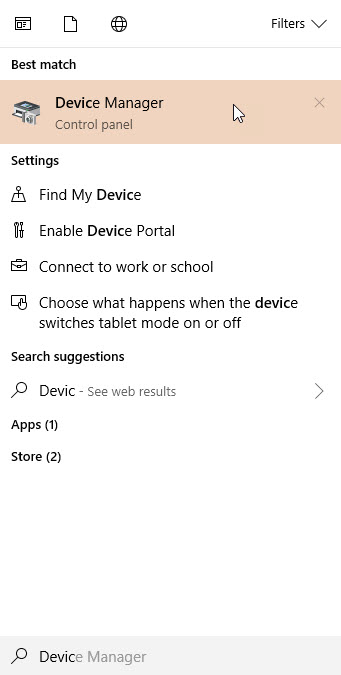


Figure . Open the Device Manager

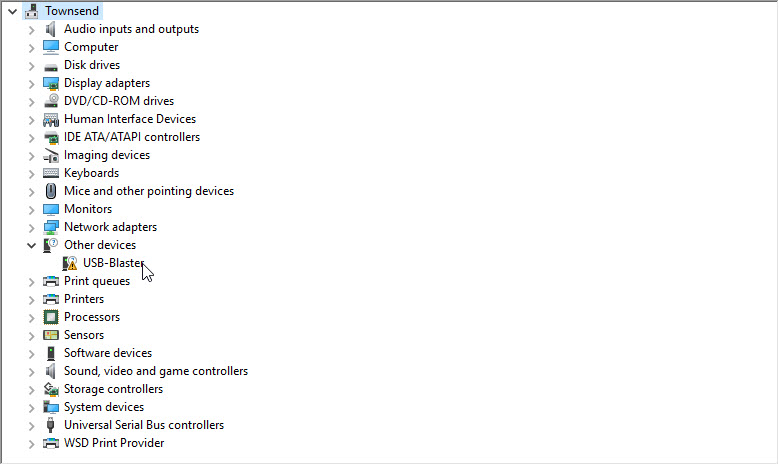


Figure . Find the Missing Driver

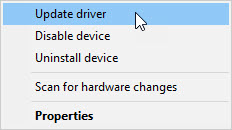


Figure . Right Click and Update Driver

Now click on ‘Browse my computer for driver software’ (Figure 50), ‘Browse’ in the next dialog (Figure 51). Now find where you installed Quartus and drill down to the ‘drivers’ folder as shown in Figure 52 and press OK. Your directory tree will look similar to, but not exactly like Figure 52; you’ll have to explore. Press ‘Next’ in the next dialog (Figure 53) and Windows will start to install the driver. When everything is done, you should see the USB Blaster driver in your Device Manager (Figure 54).

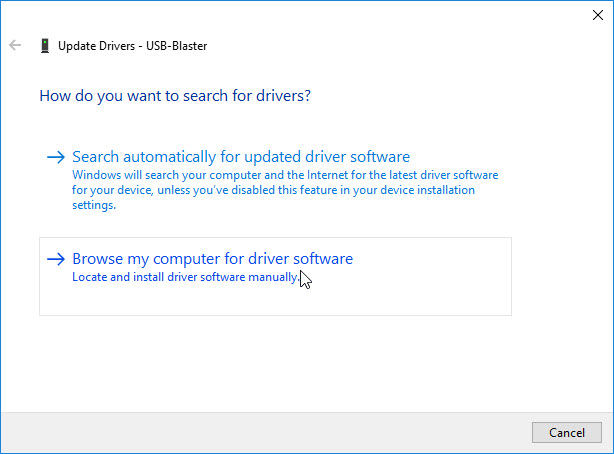


Figure . Browse My Computer

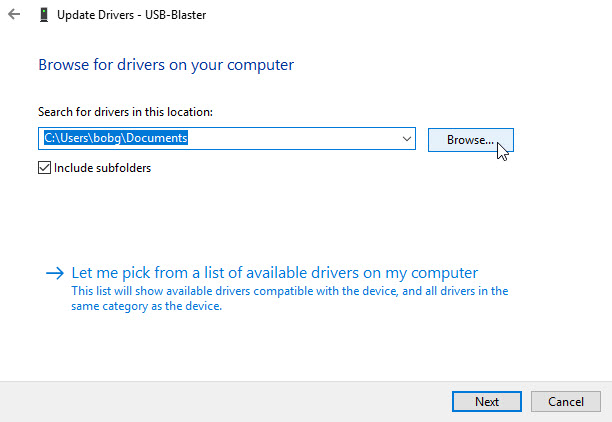


Figure . Click Browse Here (Make Sure ‘Include Subfolders’ is Checked)

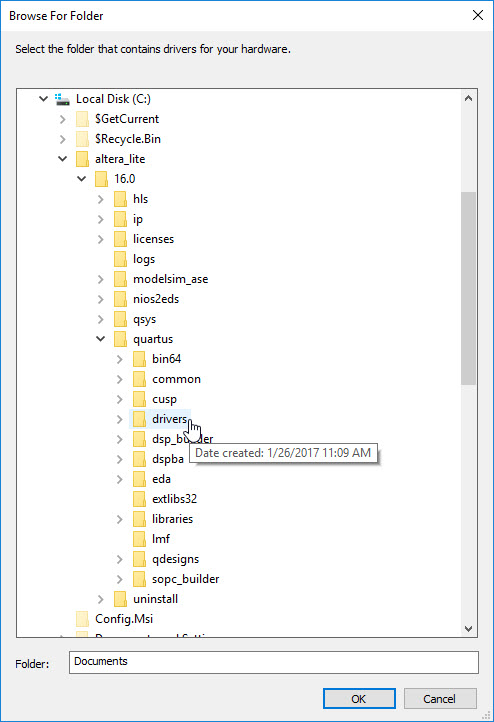


Figure . Highlight Your Quartus Drivers Folder and Press OK

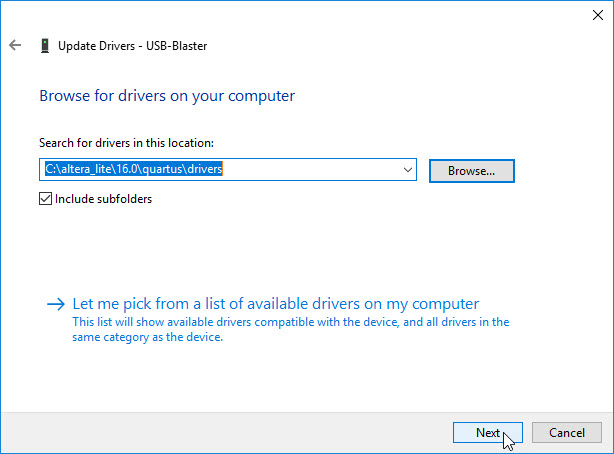


Figure . Just Press Next

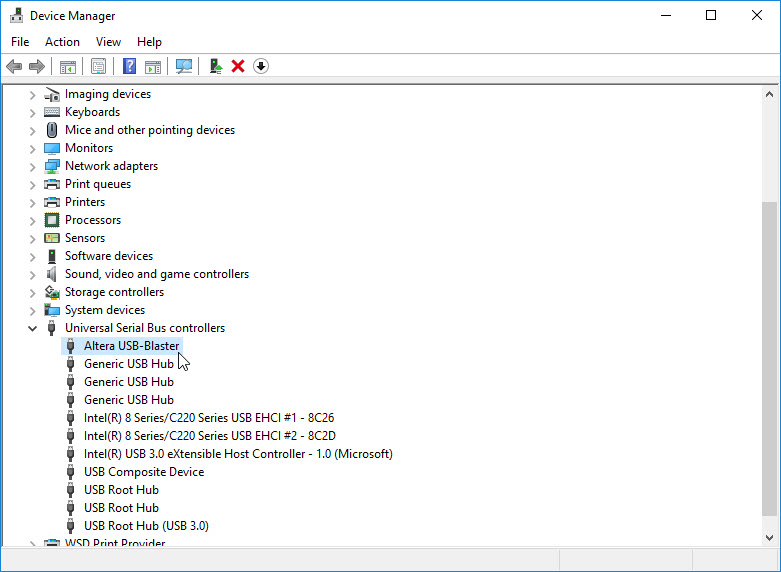


Figure . Driver Correctly Installed

Parts of this tutorial are Copyright ○c 2010 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera’s standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. This document is being provided on an “as-is” basis and as an accommodation and therefore all warranties, representations or guarantees of any kind (whether express, implied or statutory) including, without limitation, warranties of merchantability, non-infringement, or fitness for a particular purpose, are specifically disclaimed.