Quartus® Prime Introduction Using Verilog Designs

For Quartus® Prime 21.1

1 Introduction

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 software. The design process is illustrated by giving step-by-step instructions for using the Quartus Prime software to implement a very simple circuit in an Intel® (Altera) FPGA device.

The Quartus Prime system includes full support for all of the popular methods of entering a description of the desired circuit into a CAD system. Three versions of this tutorial are available; with the Verilog hardware description, with VHDL, and with schematic diagram. This tutorial makes use of the Verilog design entry method.

The last step in the design process involves programming the designed circuit into an actual FPGA device. To perform this programming step the reader needs to have an FPGA board connected to their computer, such as the DE-series Development and Education boards that are described in the `Teaching and Projects Boards` section of the FPGAcademy.org website. A reader who does not have access to a DE-series board will still find this part of the tutorial useful to learn how the FPGA programming and configuration task can be performed.

The screen captures in the tutorial were obtained using the Quartus Prime version 21.1 Standard Edition; other versions of the software may be slightly different.

2 Background

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 **??**.

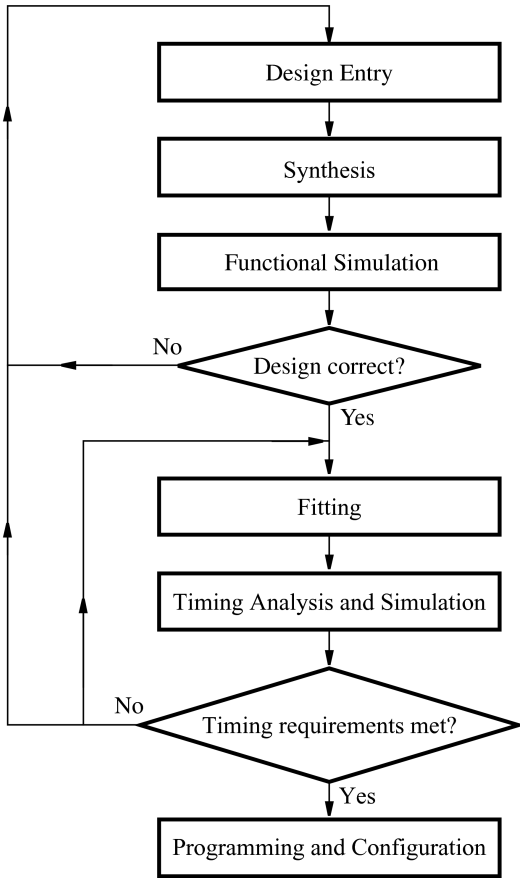


Figure 1. Typical CAD flow.

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 Prime software. It shows how the software can be used to design and implement a circuit specified by using the Verilog hardware description language. It makes use of the graphical user interface to invoke the Quartus Prime commands. Doing this tutorial, the reader will learn about:

- Creating a project
- Design entry using Verilog code
- Synthesizing a circuit specified in Verilog code
- Fitting a synthesized circuit into an FPGA
- Assigning the circuit inputs and outputs to specific pins on the FPGA
- Simulating the designed circuit
- Programming and configuring an FPGA chip

3 Getting Started

Each logic circuit, or sub-circuit, being designed with the Quartus Prime software is called a *project*. Quartus Prime 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. For this tutorial, we will use a directory named *introtutorial*. The running example for this tutorial is a simple circuit for two-way light control.

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

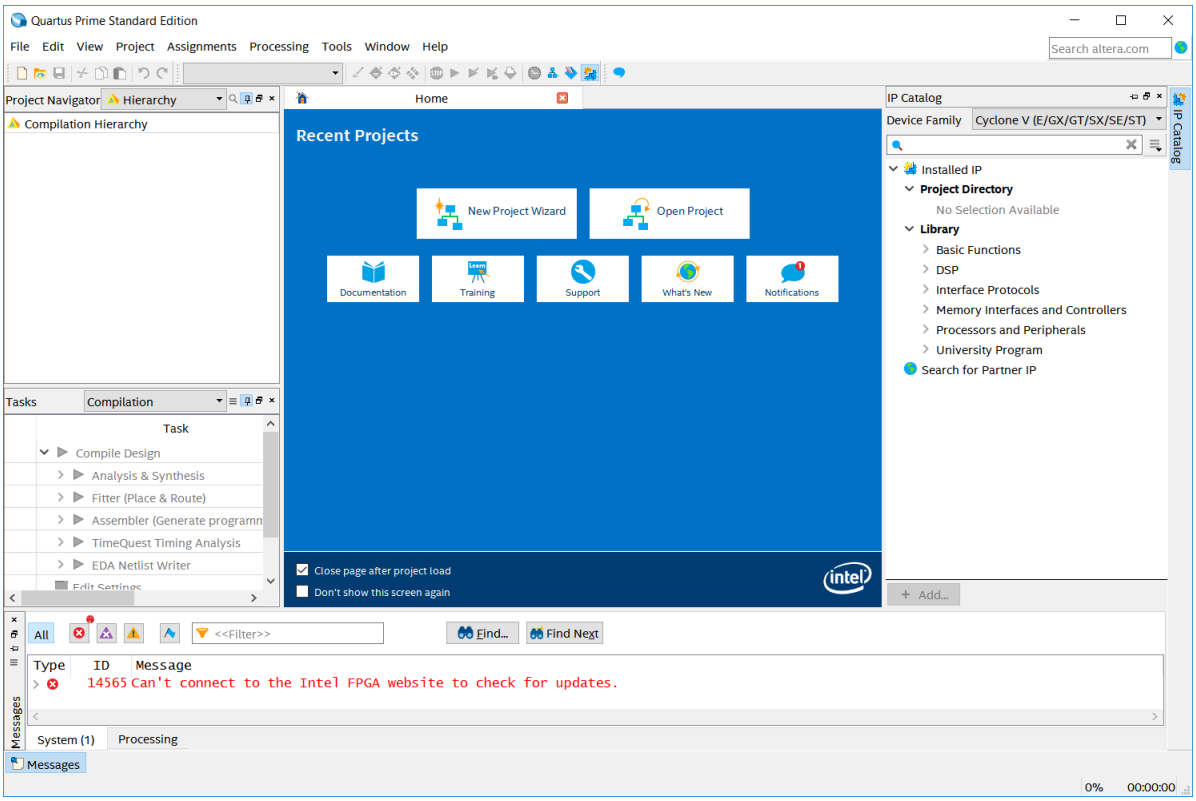


Figure 2. The main Quartus Prime display.

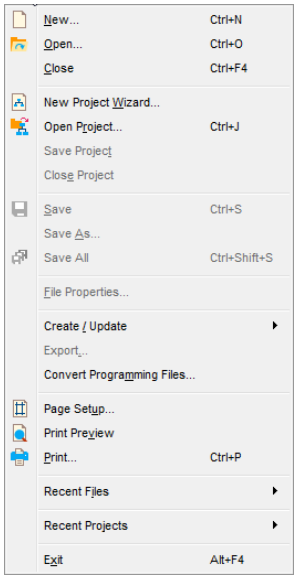


Figure 3. An example of the File menu.

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 the command name will be shown in the status bar at the bottom of the screen.

3.1 Quartus® Prime Online Help

The Quartus Prime software provides comprehensive online documentation that addresses many of the questions that may arise when using the software. The documentation is accessed from the Help menu. To get some idea of the extent of documentation provided, it is worthwhile for the reader to browse through the Help menu.

The user can quickly search through the Help topics by using the search box in the top right corner of the main Quartus display. 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.

4 Starting a New Project

To start working on a new logic circuit design we first have to define a new *project*. The 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 and click Next to reach the window in Figure **??**, which asks for the name and directory of the project.
2. Set the working directory to be *introtutorial* (of course, you can use some other directory name if preferred.) The project must have a *name*, which is usually the same as the name of 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 **??**. Press Next. If you have not yet created the directory *introtutorial*, then the Quartus Prime software will display the pop-up box in Figure **??** asking if it should create the desired directory. Clicking Yes leads to the window in Figure **??**.

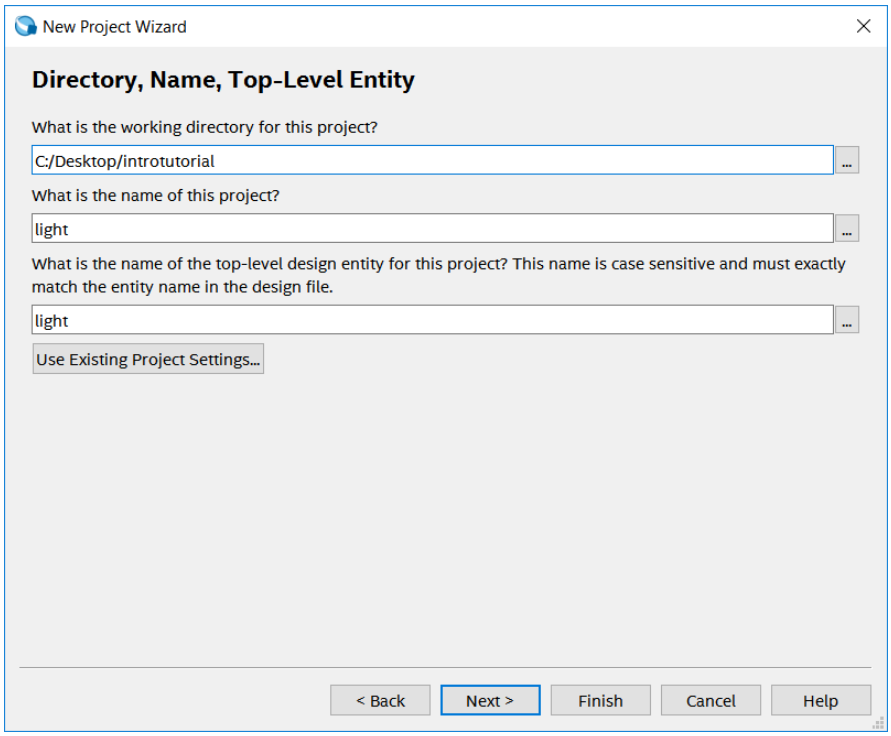


Figure 4. Creation of a new project.

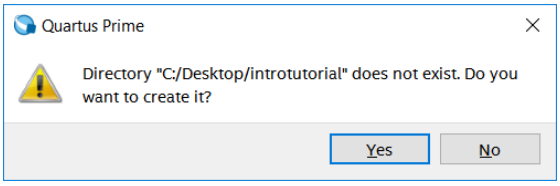


Figure 5. Quartus Prime software can create a new directory for the project.

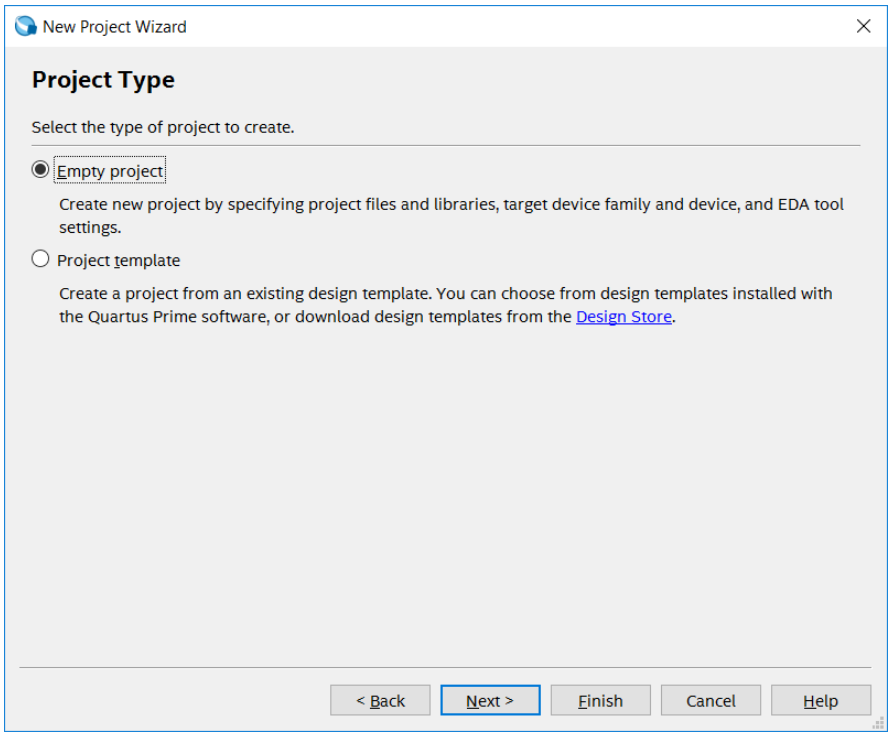


Figure 6. Choosing the project type.

3. The Project Type window, shown in Figure **??**, allows you to choose from the Empty project and the Project template options. For this tutorial, choose Empty project as we will be creating a project from scratch, and press Next which leads to the window in Figure **??**.

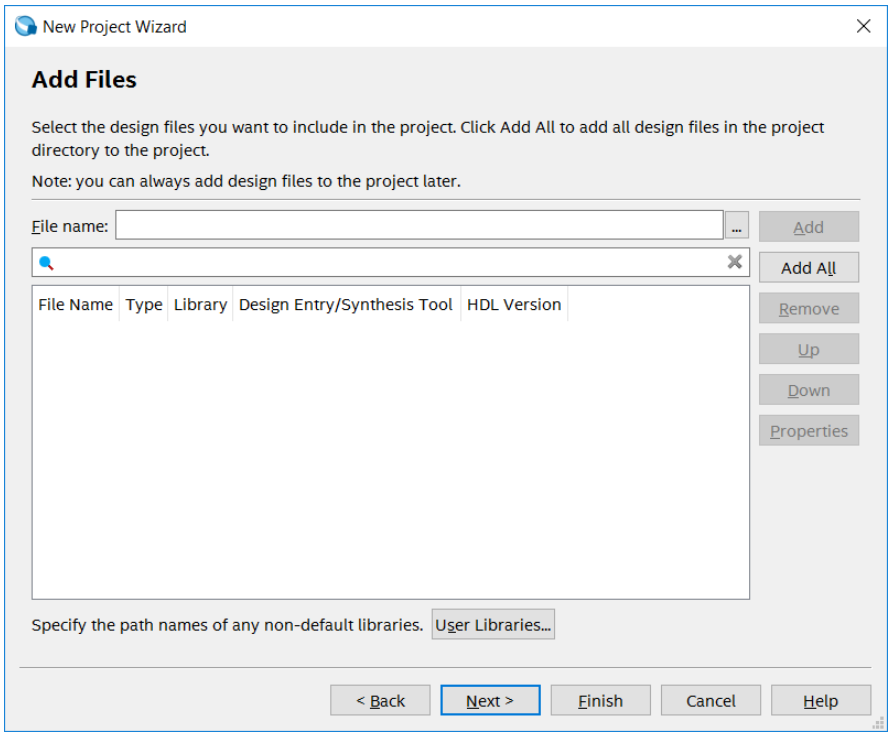


Figure 7. The wizard can include user-specified design files.

4. The wizard allows you 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 **??**.

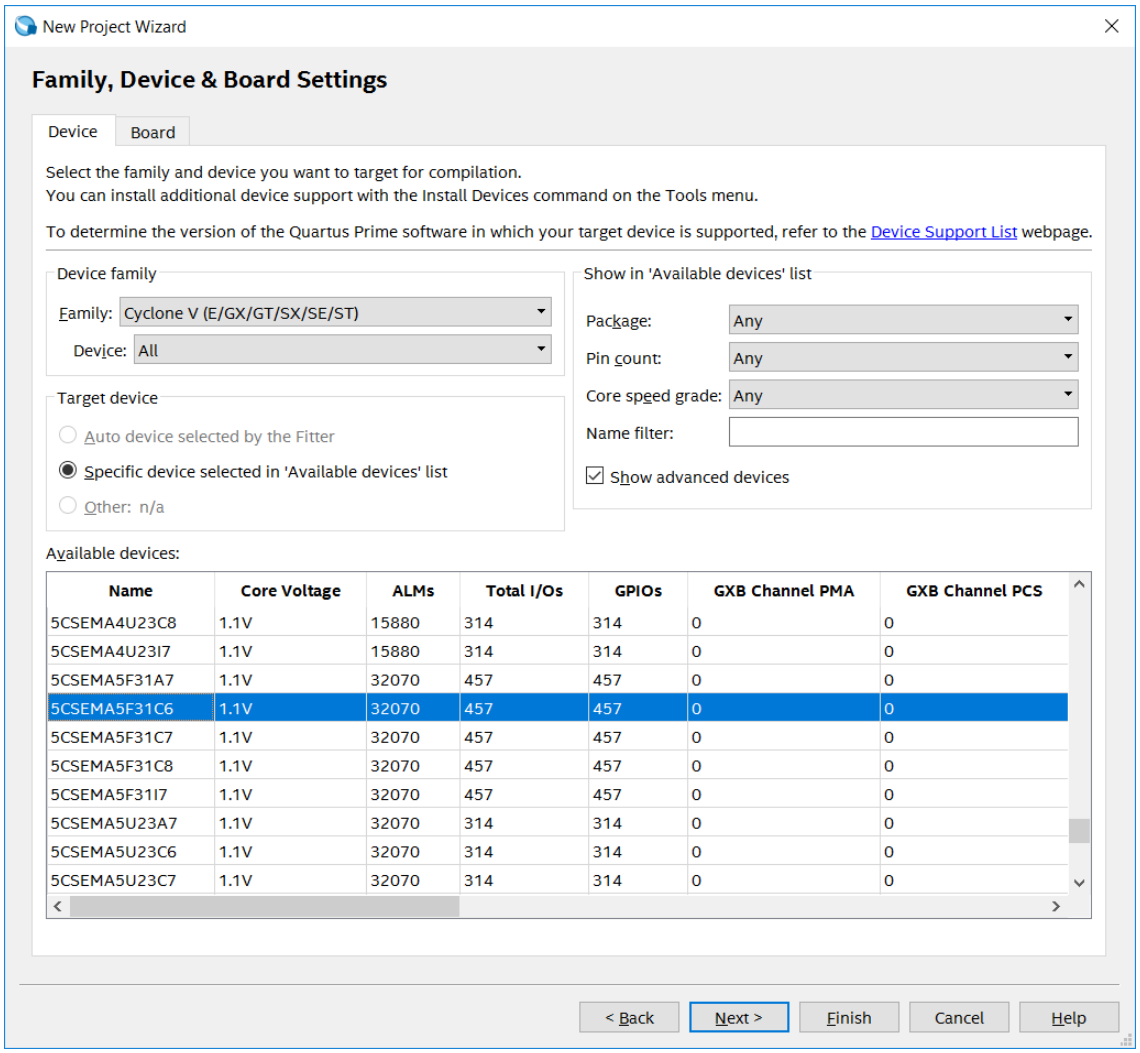


Figure 8. Choose the device family and a specific device.

5. We have to specify the type of device in which the designed circuit will be implemented. In this example we have chosen a Cyclone® series device family. We can let the Quartus Prime software select a specific device within the family, or we can choose the device explicitly. We will take the latter approach. From the list of available devices, choose the appropriate device name for your DE-series board. A list of devices names on DE-series boards can be found in Table **??**. Press Next, which opens the window in Figure **??**.

Board	Device Name
DE0-CV	Cyclone V 5CEBA4F23C7
DE0-Nano	Cyclone IVE EP4CE22F17C6
DE0-Nano-SoC	Cyclone V SoC 5CSEMA4U23C6
DE1-SoC	Cyclone V SoC 5CSEMA5F31C6
DE2-115	Cyclone IVE EP4CE115F29C7
DE10-Lite	Max 10 10M50DAF484C7G
DE10-Standard	Cyclone V SoC 5CSXFC6D6F31C6
DE10-Nano	Cyclone V SE 5CSEBA6U2317

Table 1. DE-series FPGA device names

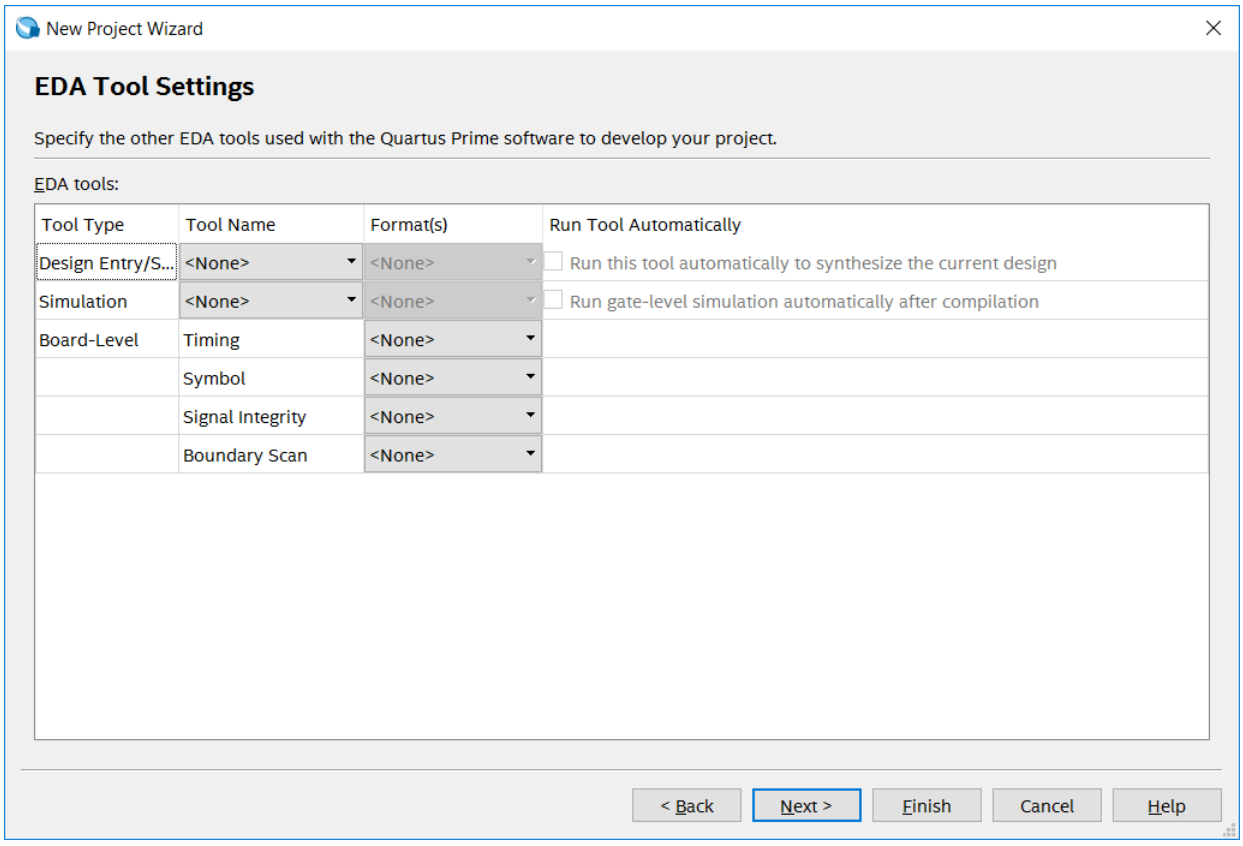


Figure 9. Other EDA tools can be specified.

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

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

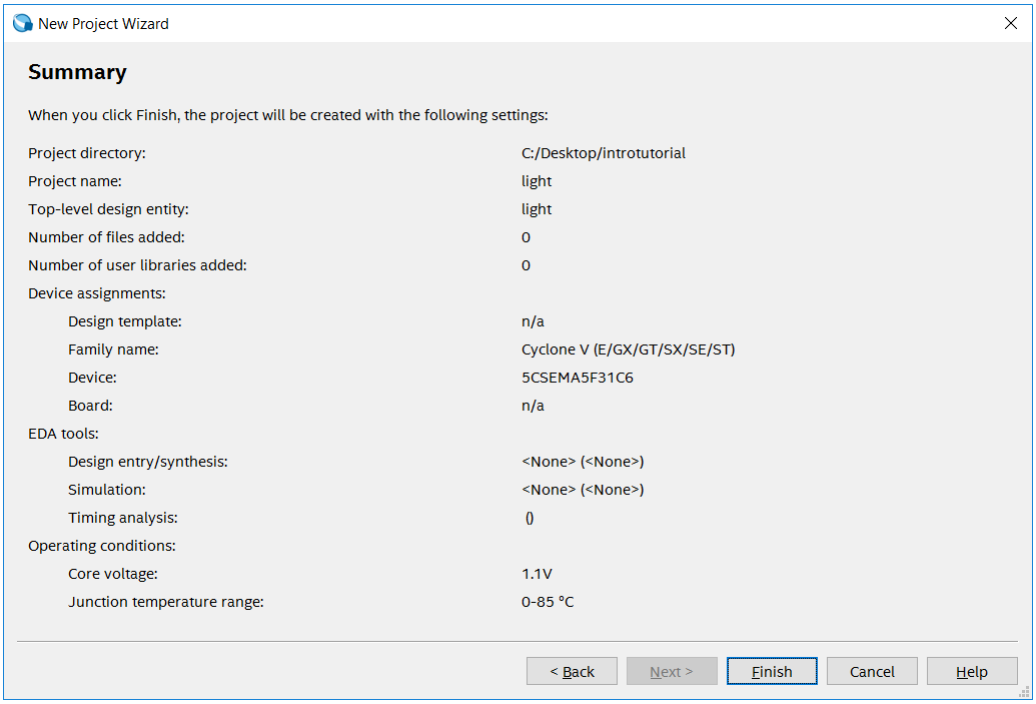


Figure 10. Summary of project settings.

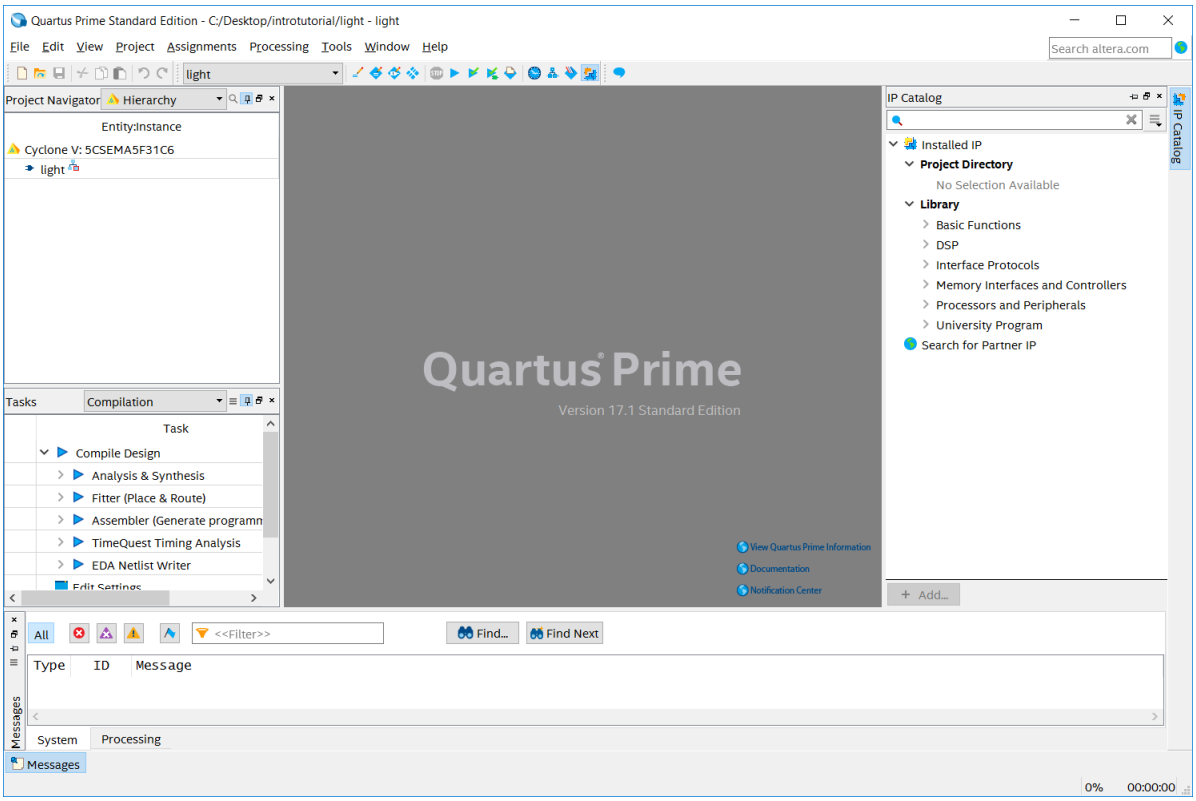
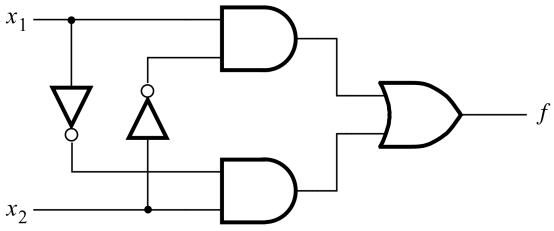


Figure 11. The Quartus Prime window for a created project.

5 Design Entry Using Verilog Code

As a design example, we will use the two-way light controller circuit illustrated in Figure **??**. This 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 specify 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 required circuit is described by the Verilog code in Figure **??**. Note that the Verilog module is called *light* to match the name given in Figure **??**, 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 *v*, which indicates a Verilog file. So, we will use the name *light.v*.

```
module light (x1, x2, f);
    input x1, x2;
    output f;
    assign f = (x1 & ~x2)|(~x1 & x2);
endmodule
```

Figure 13. Verilog code for the circuit in Figure **??**.

5.1 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.v*.

Select File > New to get the window in Figure **??**, choose Verilog HDL File, and click OK. This command 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 **??**. The box labeled Save as type should be set to Verilog HDL File. The name of the file in the box labeled File name should be *light*. Put a check-mark in the box Add file to current project. Click Save, which puts the file into the directory *introtutorial* and leads to the Text Editor window shown in Figure **??**. Enter the Verilog code in Figure **??** into the Text Editor and 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.

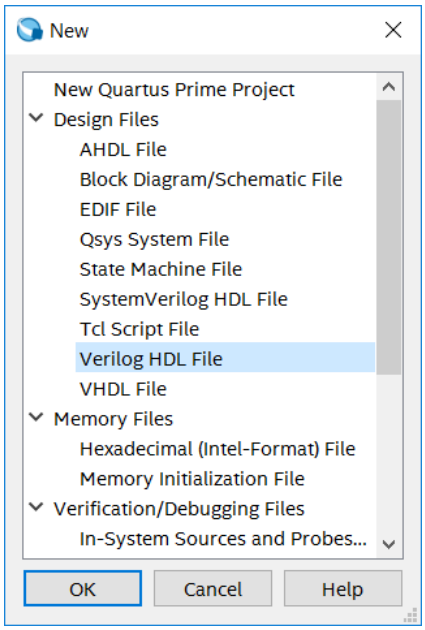


Figure 14. Choose to prepare a Verilog file.

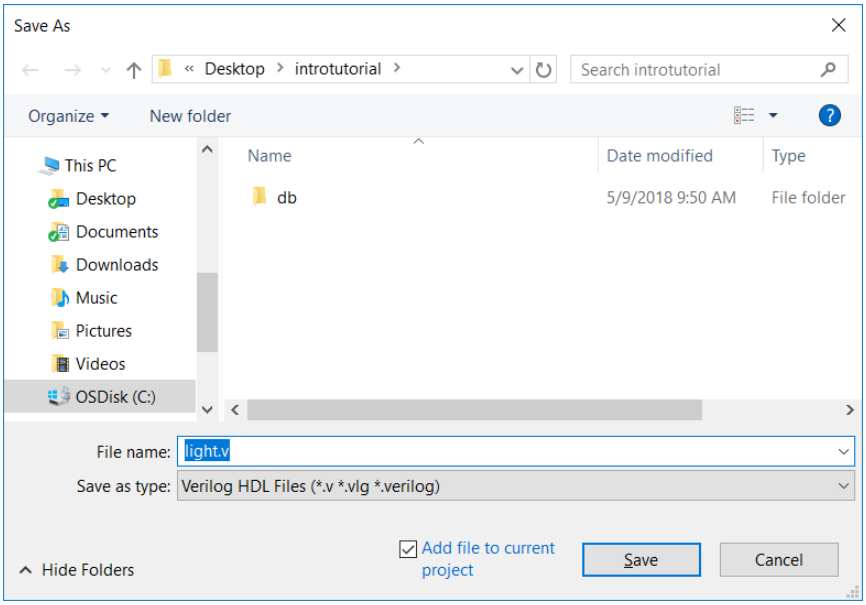


Figure 15. Name the file.

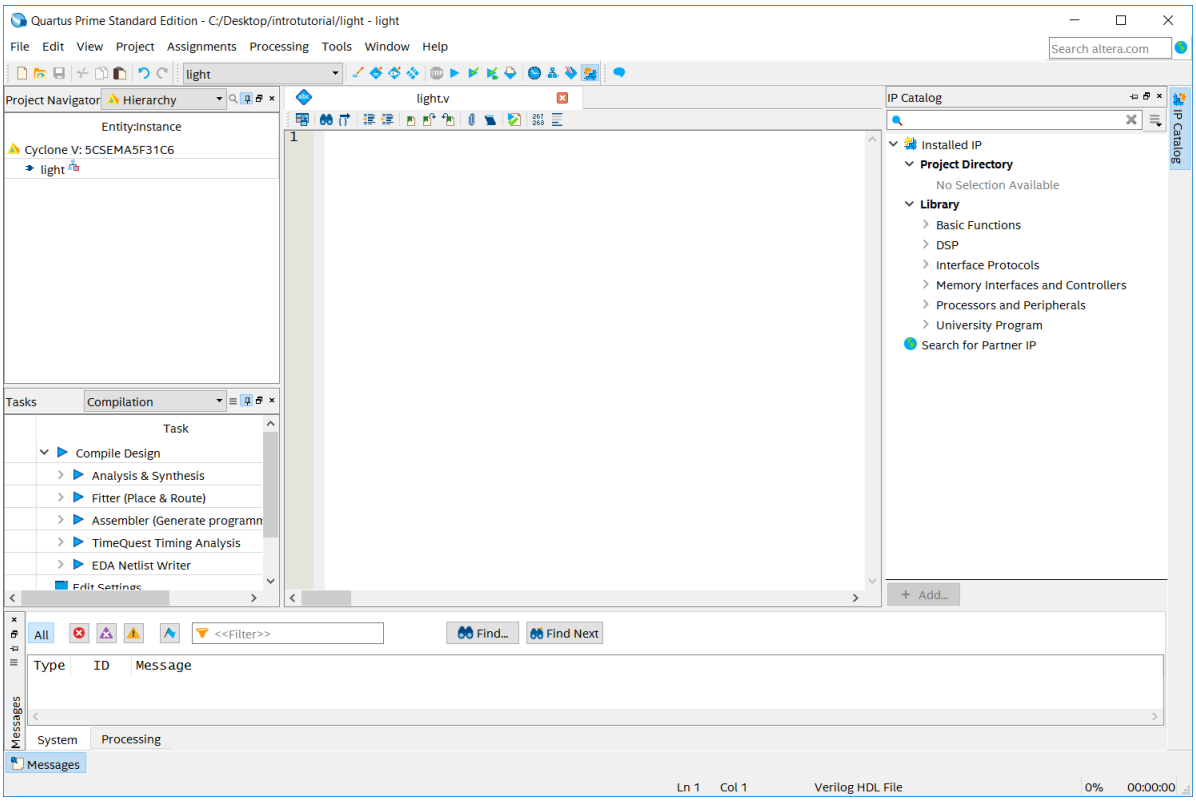


Figure 16. Text Editor window.

5.1.1 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 *Verilog templates*. The templates provide examples of various types of Verilog 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 > Verilog HDL to become familiar with this resource.

5.2 Adding Design Files to a Project

As we indicated when discussing Figure **??**, you can tell the Quartus Prime 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 **??**. 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 Prime Text Editor to create the file and checked the box labeled Add file to current project, as described in Section 5.1, then the *light.v* file is already a part of the project and will be listed in the window in Figure **??**. Otherwise, the file must be added to the project. So, if you did not use the Quartus Prime Text Editor, then place a copy of the file *light.v*, which you created using some other text editor, into the directory *introtutorial*. To add this file to the project, click on the ... button next to the box labeled File name in Figure **??** to get the pop-up

window in Figure **??**. Select the *light.v* file and click Open. The selected file is now indicated in the File name box in Figure **??**. Click Add then OK to include the *light.v* file in the project. We should mention that in many cases the Quartus Prime software is able to automatically find the right files to use for each entity referenced in Verilog code, even if the file has not been explicitly added to the project. However, for complex projects that involve many files it is a good design practice to specifically add the needed files to the project, as described above.

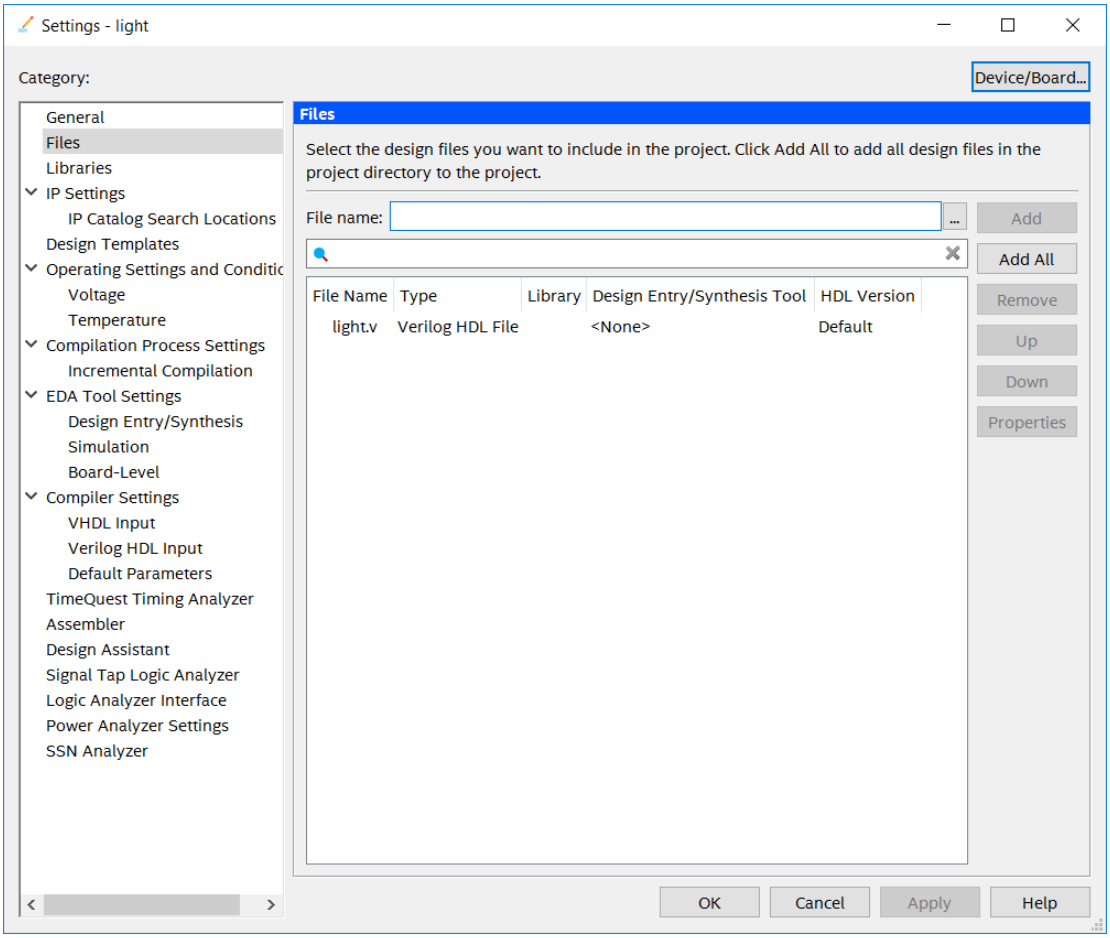


Figure 17. Settings window.

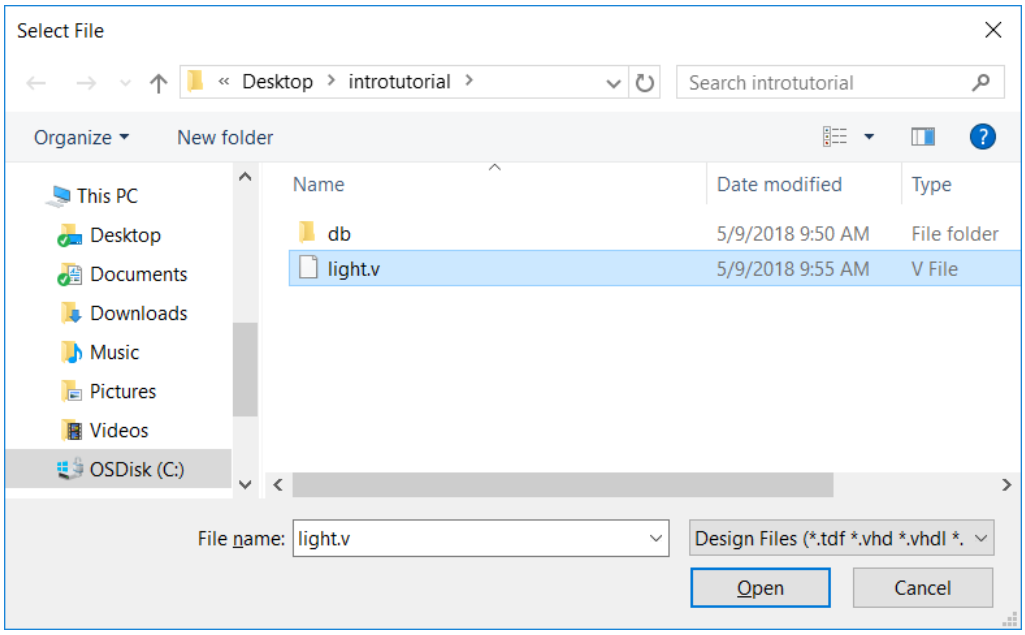


Figure 18. Select the file.

6 Compiling the Designed Circuit

The Verilog code in the file *light.v* is processed by several Quartus Prime 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. Your project must be saved before compiling. As the compilation moves through various stages, its progress is reported in a window on the left side of the Quartus Prime display. In the message window, at the bottom of the figure, various messages are displayed throughout the compilation process. In case of errors, there will be appropriate messages given.

When the compilation is finished, a compilation report is produced. A tab showing this report is opened automatically, as seen in Figure **??**. This tab can be 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. Figure **??** 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.

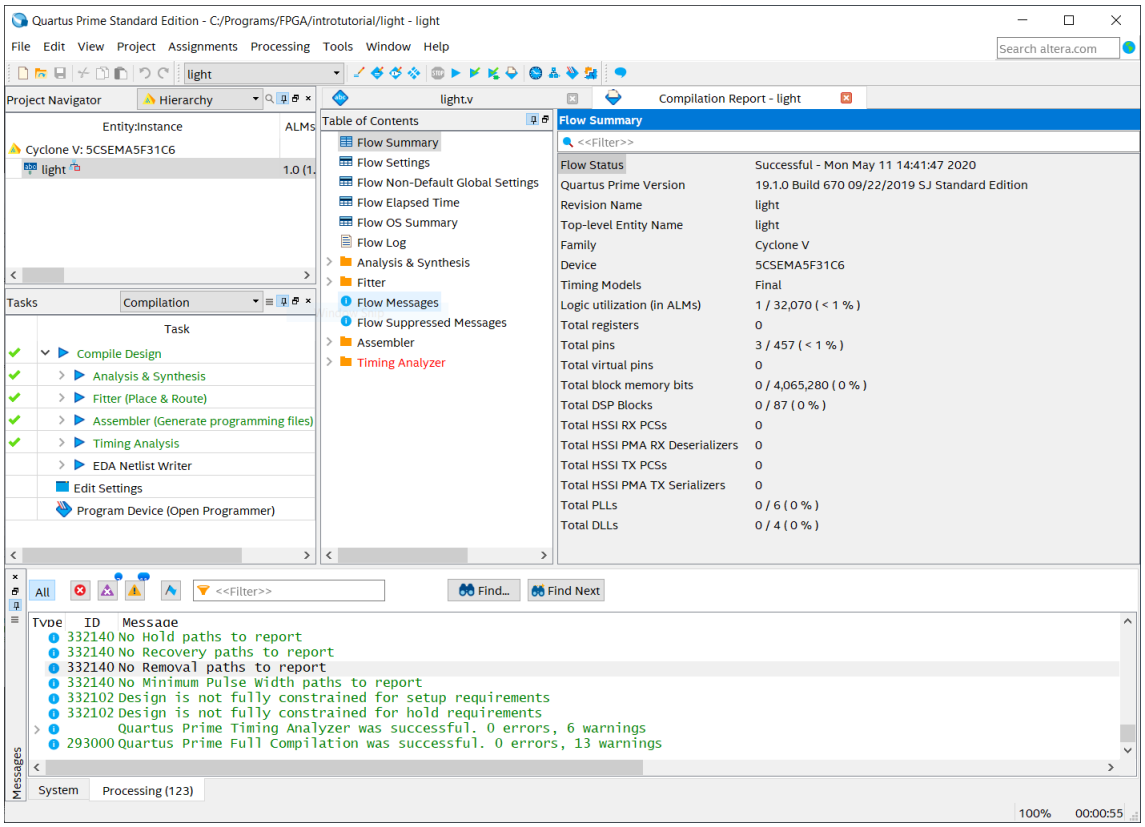


Figure 19. Display after a successful compilation.

6.1 Errors

The Quartus Prime software displays messages produced during compilation in the Messages window. If the Verilog design file has been correctly specified, then 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 Verilog 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 Verilog code in the Text 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.v*. Remove the semicolon in the **assign** statement, illustrating a common typographical error. Compile the erroneous design file by clicking on the  icon. A pop-up box will ask if the changes made to the *light.v* file should be saved; click Yes. After trying to compile the circuit, Quartus Prime software will display error messages in the Messages window, and show that the compilation failed in the Analysis & Synthesis stage of the compilation process. The compilation report summary, given in Figure **??**, confirms the

failed result. In the Table of Contents panel, expand the Analysis & Synthesis part of the report and then select Messages to have the messages displayed as shown in Figure **??**. The Compilation Report can be displayed as a separate window as in Figure **??** by right-clicking its tab and selecting Detach Window, and can be reattached by clicking Window > Attach Window. Double-click on the first error message. The Quartus Prime software responds by opening the *light.v* file and highlighting the statement which is affected by the error, as shown in Figure **??**. Correct the error and recompile the design.

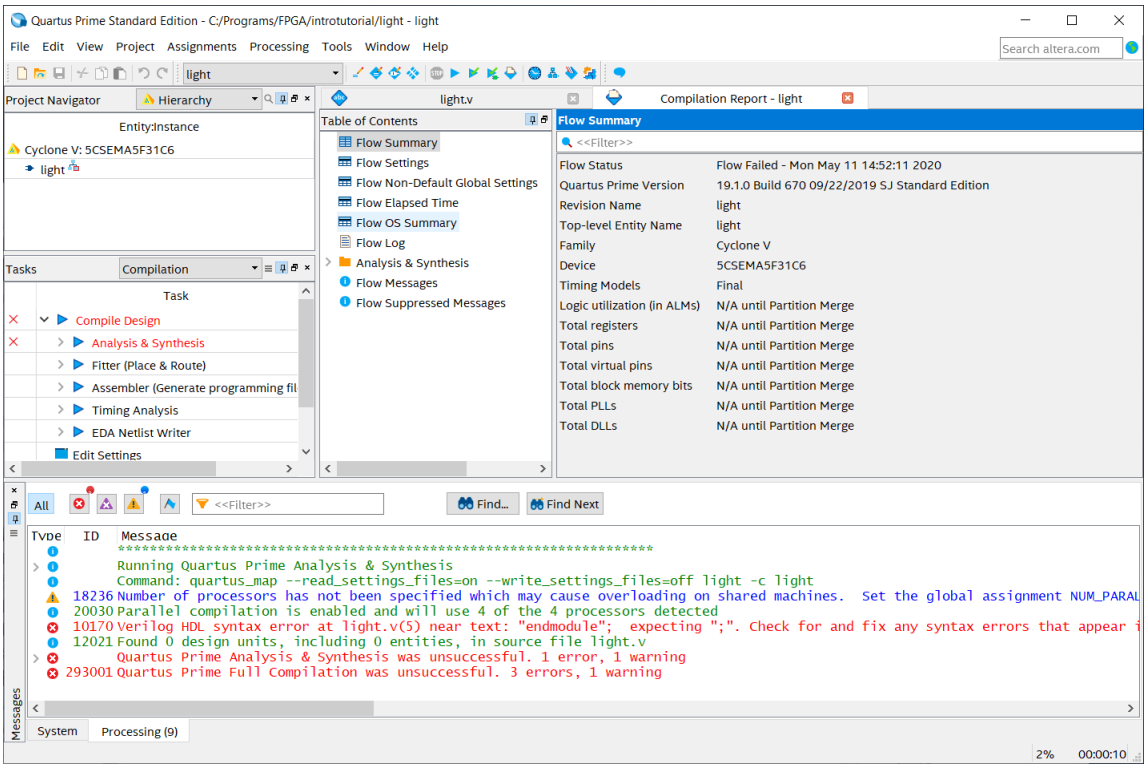


Figure 20. Compilation report for the failed design.

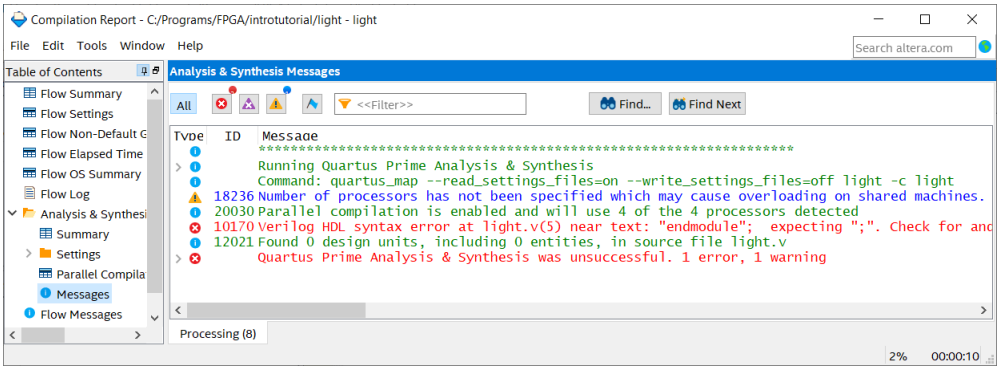


Figure 21. Error messages.

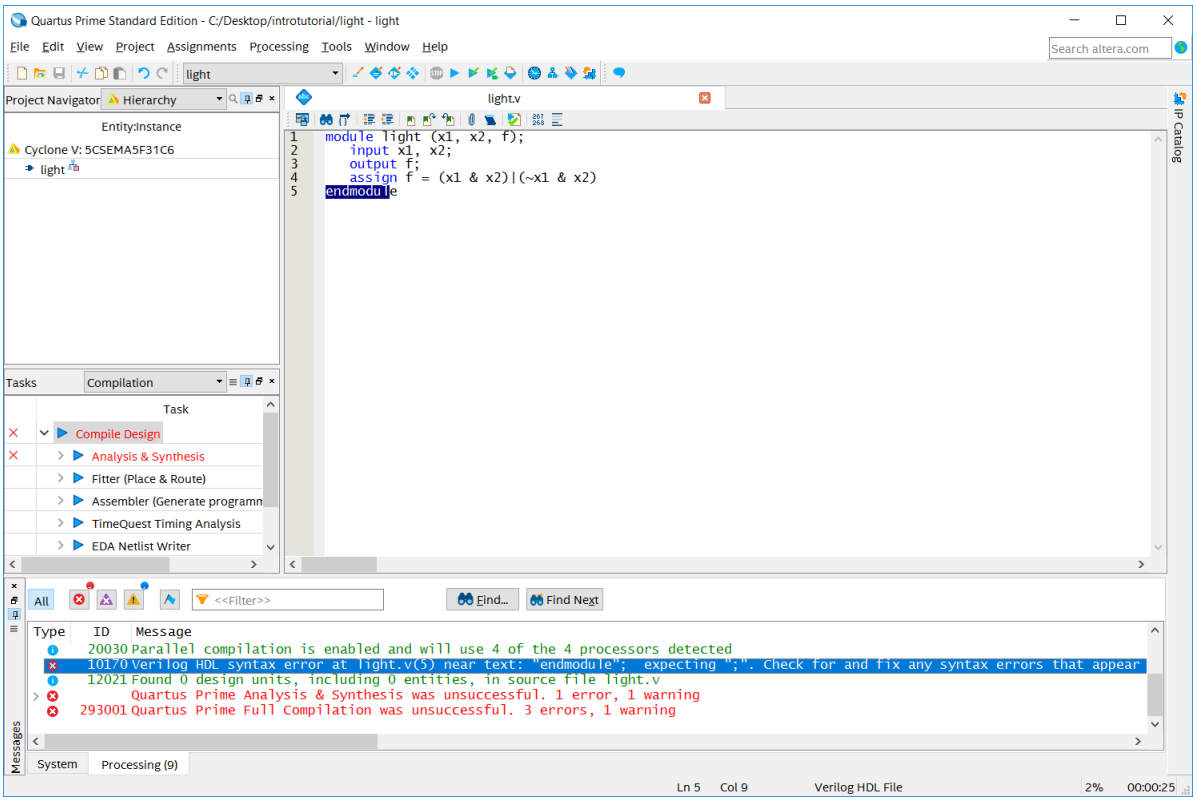


Figure 22. Identifying the location of the error.

7 Pin Assignment

During the compilation process described above, the Compiler was free to choose any pins on the selected FPGA device to serve as inputs and outputs. However, an FPGA board has hardwired connections between the pins of the FPGA chip and the other components that are 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 listed in Table **??**. We will connect the output f to a light-emitting diode on your DE-series board. For the DE2-115 we will use a green LED: $LEDG_0$. On the DE0-CV, DE1-SoC, DE10-Lite and DE10-Standard we will use $LEDR_0$. On the DE0-Nano and DE0-Nano-SoC, we will use LED_0 The FPGA pin assignment for the LEDs are listed in Table **??**.

Component	SW_0	SW_1	$LEDG_0$, LED_0, or $LEDR_0$
DE0-CV	PIN_U13	PIN_V13	PIN_AA2
DE0-Nano	PIN_M1	PIN_T8	PIN_A1
DE0-Nano-SoC	PIN_L10	PIN_L9	PIN_W15
DE2-115	PIN_AB28	PIN_AC28	PIN_E21
DE1-SoC	PIN_AB12	PIN_AC12	PIN_V16
DE10-Lite	PIN_C10	PIN_C11	PIN_A8
DE10-Standard	PIN_AB30	PIN_AB28	PIN_AA24
DE10-Nano	PIN_Y24	PIN_W24	PIN_W15

Table 2. DE-Series Pin Assignments

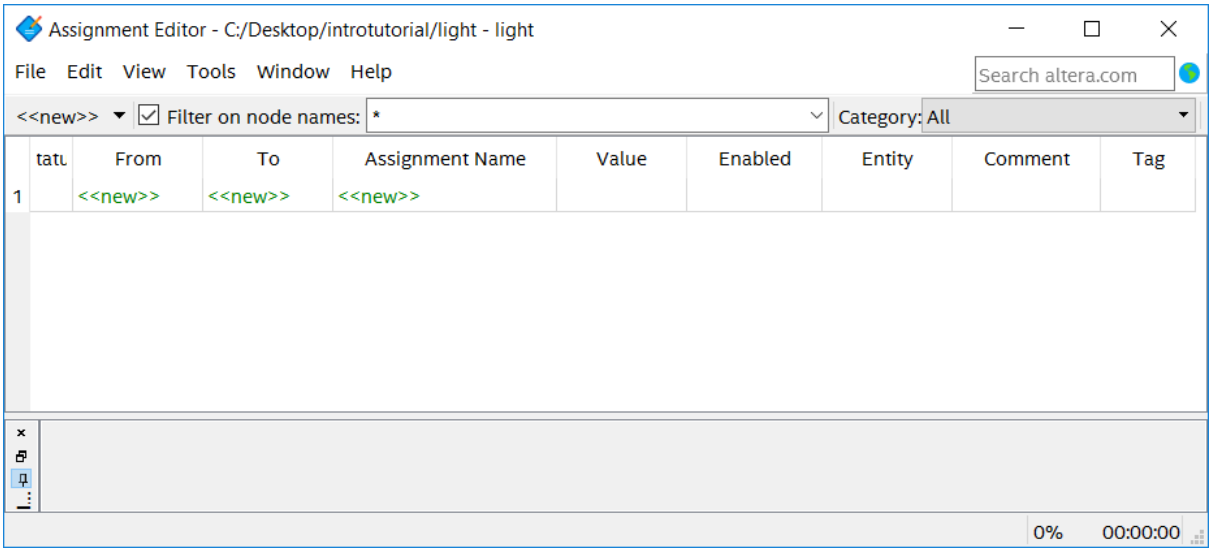


Figure 23. The Assignment Editor window.

Pin assignments can be made by using the *Assignment Editor*. Select Assignments > Assignment Editor to reach the window in Figure **??** (shown here as a detached window). In the Category drop-down menu select All. Click on the <<new>> button located near the top left corner to make a new item appear in the table. Double-click the box under the column labeled To so that the Node Finder button appears. Click on the button (not the drop down arrow) to reach the window in Figure **??**. Click on and to show or hide more search options. In the Filter drop-down menu select Pins: all. Then, click the List button to display the input and output pins to be assigned: f, $x1$, and $x2$. Click on $x1$ as the first pin to be assigned and click the > button; this action will enter $x1$ in the Nodes Found box. Click OK. The $x1$ node will now appear in the box under the column labeled To in the Assignment Editor window. Alternatively, the node name can be entered directly by double-clicking the box under the To column and typing in the node name.

Now double-click on the box to the right of this new $x1$ entry, in the column labeled Assignment Name, to open the drop-down menu in Figure **??**. Scroll down and select Location (Accepts wildcards/groups). Instead of scrolling

down the menu to find the desired item, you can alternatively type the first letter of the item in the Assignment Name box. In this case the desired item happens to be the first item beginning with L. Finally, double-click the box in the column labeled Value. Type the pin assignment corresponding to SW_0 for your DE-series board, as listed in Table **??**.

Use the same procedure to assign input $x2$ and output f to the appropriate pins listed in Table **??**. An example using a DE1-SoC board is shown in Figure **??**. 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. Finally, execute the Processing > Start Compilation command to create a new circuit that makes use of your pin assignments.

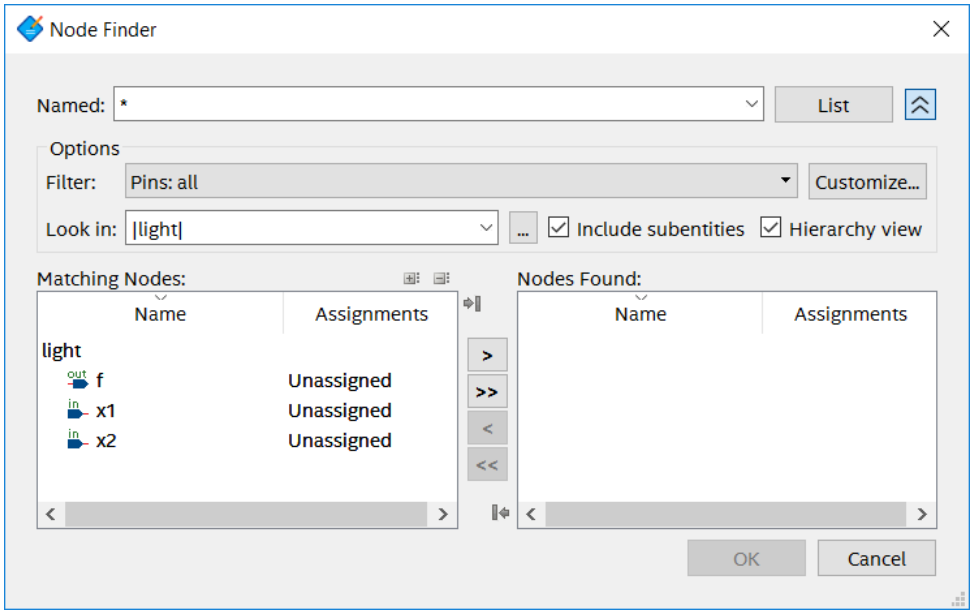


Figure 24. The Node Finder displays the input and output names.

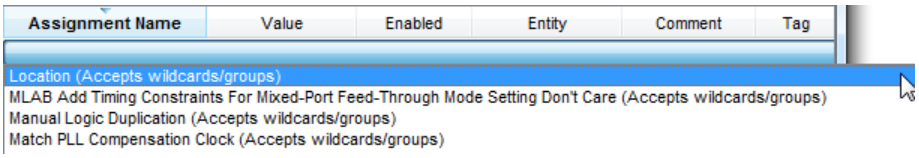


Figure 25. The available assignment names for a DE-series board.

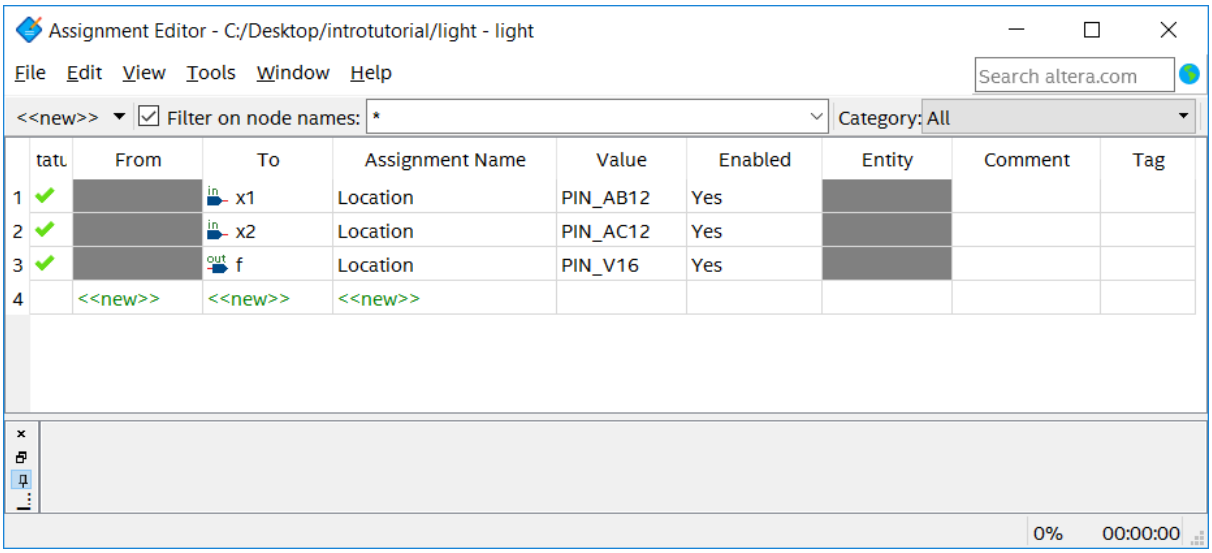


Figure 26. The complete assignment.

When you make assignments related to your project, such as the pin assignments described above, or the device assignments made earlier in the tutorial, Quartus Prime stores these assignments in a special type of file associated with your project. This file is called a *quartus settings file* (*qsf*), and in our case would be named *light.qsf*. In this file the pin assignments created above (using a DE1-SoC board as an example) would be included as

```
set_location_assignment PIN_AB12 -to x1
set_location_assignment PIN_AC12 -to x2
set_location_assignment PIN_V16 -to f
```

A useful Quartus Prime feature allows the user to import into a project the assignments contained in a quartus settings file, rather than creating them manually using the Assignment Editor. Importing pin assignments from an existing *qsf* file can be a convenient way of making these assignments, rather than following the (somewhat tedious) process described above.

You can import pin assignments by choosing Assignments > Import Assignments. This command opens the dialogue in Figure **??**, allowing you to select a file to import.

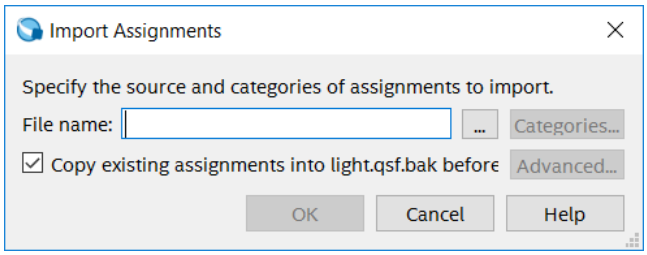


Figure 27. Importing the pin assignment.

For convenience, all important settings for a specific DE-series board, such as device and pin assignments, are available in a file that can be downloaded from the `Teaching and Projects Boards` section of the website FPGAcademy.org. Any of these assignment files can be downloaded to your computer by clicking on the corresponding QSF item that is provided along with the description of the board.

As an example, the QSF file that you can download for the DE1-SoC board is named *DE1_SoC.qsf*. This file provides various types of assignments, including pin assignments for *all* signals on the board. You can import these assignments into your Quartus Prime project, as described above. The pin assignments in the file *DE1_SoC.qsf* use the signal names that are defined in the user manual for the board, in this case the *DE1-SoC 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 Verilog code that are used in this *DE1_SoC.qsf* file. In our example, this means that instead of the top-level ports named *x1*, *x2* and *f*, we would name these ports *SW[0]*, *SW[1]* and *LEDG[0]*, respectively, because those are the names used in the *DE1_SoC.qsf* file.

8 Programming and Configuring the FPGA Device

The FPGA device must be programmed to implement the designed circuit. The required configuration file is generated by the Quartus Prime Compiler's Assembler module. Each DE-series 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 Prime software) to the board by means of a cable that connects a USB port on the host computer to the *USB-Blaster* connector on the board. To use this connection, it is necessary to have the USB-Blaster (II) driver installed. If this driver is not already installed, consult the tutorial *Getting Started with the Terasic DE-Series Boards*, available on FPGAcademy.org, for information about installing the driver. You can also search on the Internet for a topic such as "Installing USB Blaster driver". Before using the board, make sure that the USB cable is properly connected and that the board is powered on.

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 Prime software places the configuration data into the configuration device on the DE-series board. Then, this data is loaded into the FPGA upon power-up or reconfiguration. Thus, the

FPGA need not be configured by the Quartus Prime software if the power is turned off and on. The choice between the two modes is made by switches on the DE-series board. Consult your manual for the location of this switch on your DE-series board. The boards should be set to JTAG mode by default. This tutorial discusses only the JTAG programming mode.

8.1 JTAG* Programming for the DE1-SoC Board, DE0-Nano-SoC, DE10-Nano, and DE10-Standard

For the DE1-SoC Board, DE0-Nano-SoC, DE10-Nano, and DE10-Standard boards, the following steps should be used for programming. Select Tools > Programmer to reach the window in Figure **??**. If the *SOCVHPS* device shown in the figure is missing, then you need to add it. Click on the Add Device menu, and then look for the *SOCVHPS* device under the *Soc Series V* family. Now 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. The required programming hardware is called *DE-SoC*. If this hardware is not chosen by default as the programming hardware, then press the Hardware Setup... button and select the *DE-SoC* in the window that pops up, as shown in Figure **??**.[1]

Observe that the configuration file *light.sof* in directory *output_files* is listed in the window in Figure **??**. 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. Ensure the Program/Configure box is checked. This setting is used to select the FPGA in the Cyclone V SoC chip for programming.

Press Start in the Programmer to program the FPGA device. An LED on the board will light up while the FPGA device is being programmed. If you see an error reported by Quartus Prime software indicating that programming *failed*, then check to ensure that the board is properly powered on. Also, as mentioned above, if the *SOCVHPS* device is not shown as in Figure **??**, click Add Device > SoC Series V > SOCVHPS, and then click OK to add it. Some boards use the device order shown in Figure **??**, and others use the opposite order. If programming of your board failed, try reversing the order from what is shown in the figure, by clicking on a device and then clicking 

or 

.

[1] In some versions of the Microsoft Windows operating system the drivers provided with older versions of the Quartus Prime software might fail to install. In such cases, no programming Hardware will be available to the Quartus Programmer. One approach to solving such an issue is to download and install the drivers that are provided with a more recent version of the Quartus Prime software. If you install a version of Quartus Prime into a folder named `Qdir`, then the drivers that come with this version can be found in the sub-folder `Qdir/quartus/drivers`. The driver for the DE-SoC hardware is called *USB-Blaster-II*.

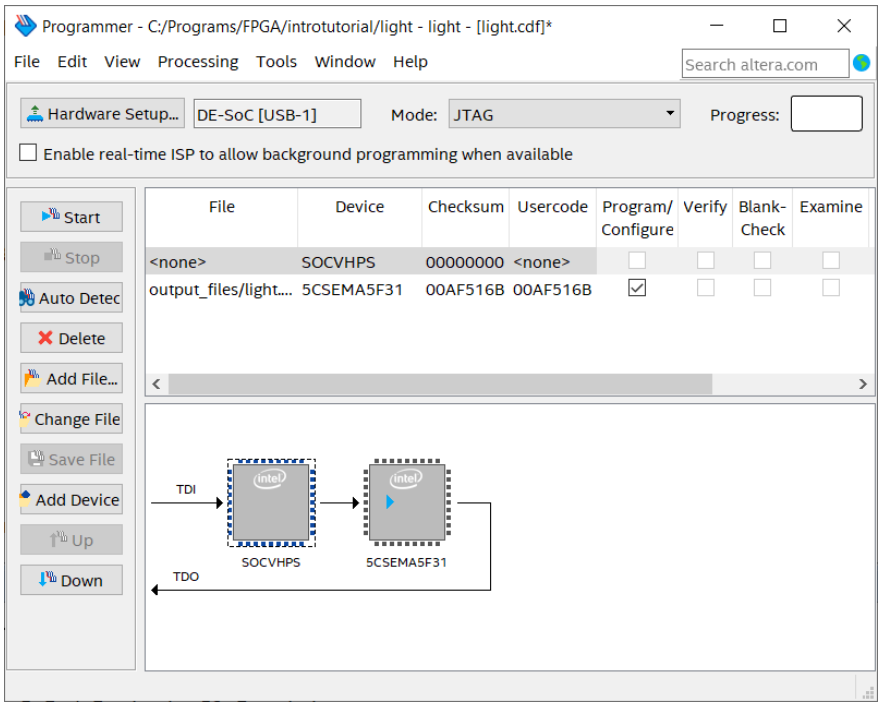


Figure 28. The Programmer window.

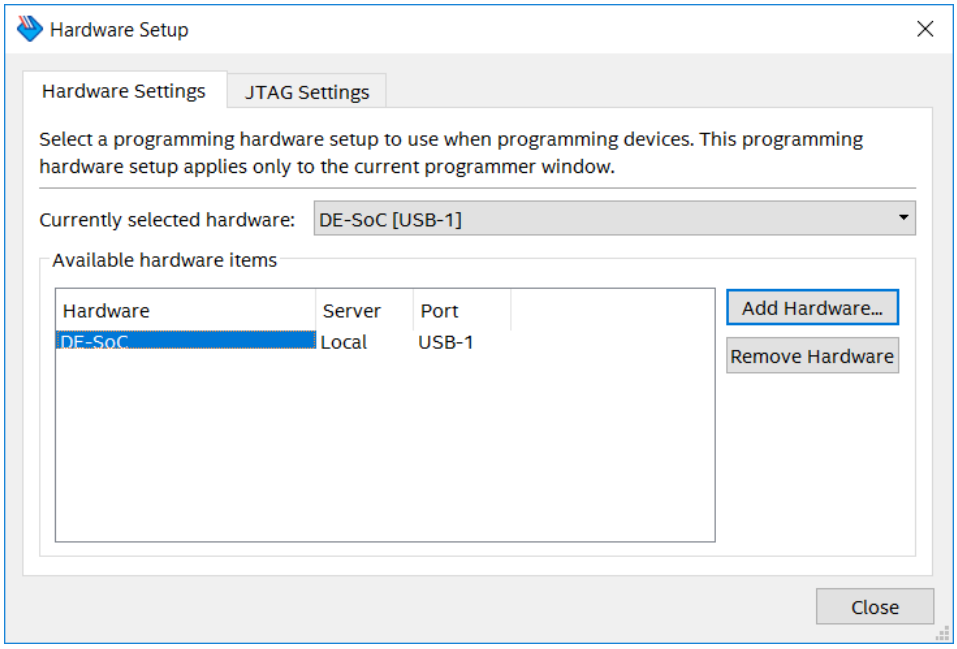


Figure 29. The Hardware Setup window.

8.2 JTAG* Programming for the DE0-CV, DE0-Nano, DE10-Lite, and DE2-115 Boards

For the DE0-CV, DE0-Nano, DE10-Lite, and DE2-115 Boards, the programming and configuration task is performed as follows. Select Tools > Programmer to reach the window in Figure **??**. 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 **??**. [2]

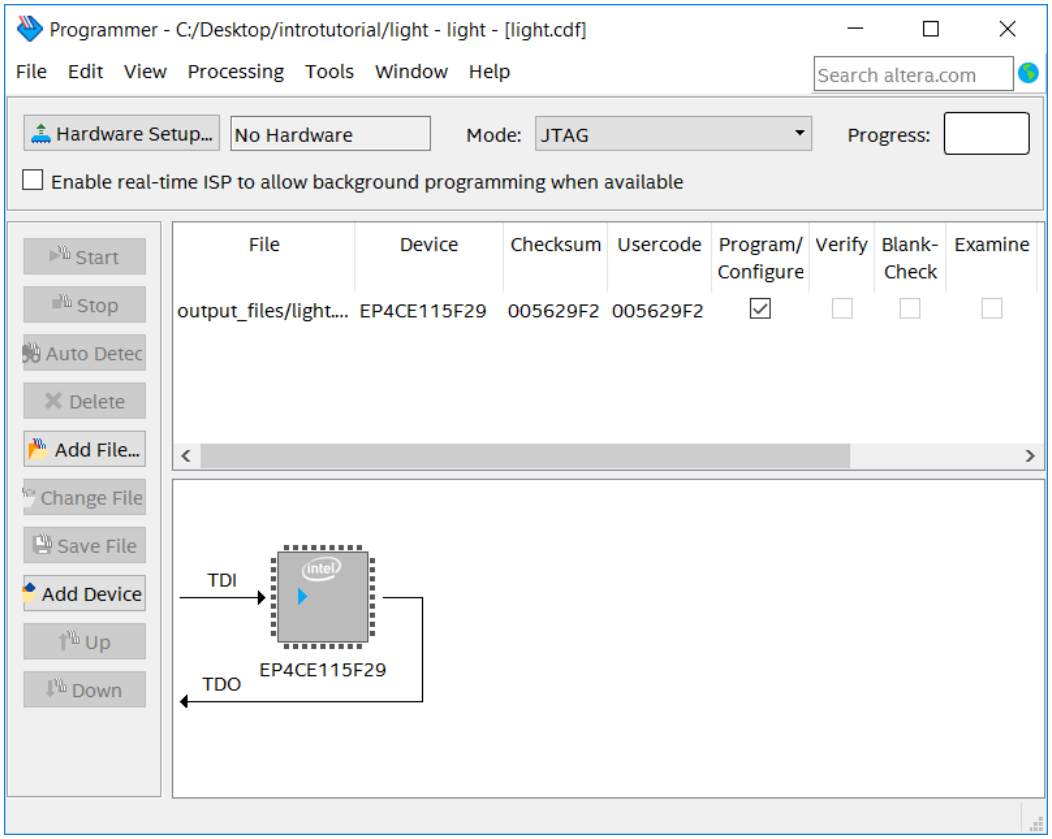


Figure 30. The Programmer window.

Observe that the configuration file *light.sof* is listed in the window in Figure **??**. 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. Ensure the Program/Configure check box is ticked, as shown in Figure **??**.

[2] In some versions of the Microsoft Windows operating system the USB-Blaster driver provided with older versions of the Quartus Prime software might fail to install. In such cases, no programming Hardware will be available to the Quartus Programmer. One approach to solving such an issue is to download and install the USB-Blaster driver that is provided with a more recent version of the Quartus Prime software. If you install a version of Quartus Prime into a folder named `Qdir`, then the drivers that come with this version can be found in the sub-folder `Qdir/quartus/drivers`.

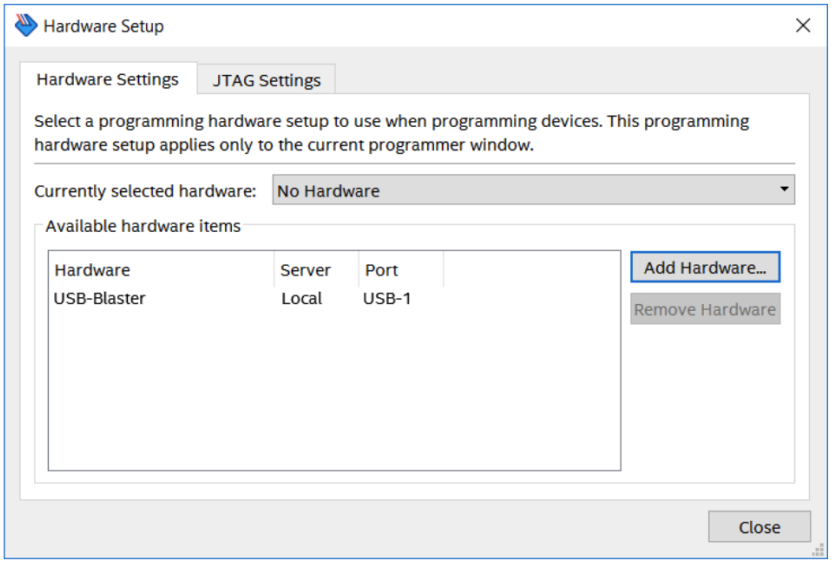


Figure 31. The Hardware Setup window.

Now, press Start in the window in Figure **??**. An LED on the board will light up corresponding to the programming operation. If you see an error reported by Quartus Prime software indicating that programming failed, then check to ensure that the board is properly powered on.

9 Testing the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE-series board, it is prudent to simulate it to ascertain its correctness. While not covered in this tutorial, users may use software such as *ModelSim* or other simulation environments to test the circuit in simulation. Simulation of a circuit often provides a comprehensive view of the circuit's functionality, and can help users easily find bugs within the circuit's logic without having to touch hardware.

Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. 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. Verify that the circuit implements the truth table in Figure **??**.

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