
appendix**D****TUTORIAL 3—PHYSICAL
IMPLEMENTATION IN AN FPGA**

In this tutorial we focus on the physical implementation of a design project in an FPGA chip. We show how to manually choose which pins on a device package are used for the input and output signals in a circuit, and we describe how to use the Quartus II Programmer module to transfer a compiled circuit into the selected FPGA chip.

D.1 MAKING PIN ASSIGNMENTS

In the examples given in Tutorial 2, the assignment of input and output signals to device pins was done automatically by the Compiler. In some cases the designer needs to be able to manually specify which pins to use for some of the signals in a circuit. For example, the circuit board that contains the FPGA chip will have hardwired connections from some of the FPGA pins to other components, such as switches or LEDs. To make use of the hardwired connections, the designer has to be able to specify which device pins should be used for a particular design.

To assign pins manually, it is necessary to specify which chip to use. This was already done in section C.1.1, when we selected the EP2C35F672C6 FPGA as shown in Figure C.2. Open again the project *example_verilog*, which was done in section C.1.

In section C.1.4 we used the Chip Planner to examine the compilation results for the *example_verilog* circuit. As depicted in Figures C.6 and C.7 the Chip Planner shows the FPGA's I/O cells, often called *pads*, which are arranged around the periphery of the chip. To see how these pads correspond to pins on the FPGA chip package, we can use the Pin Planner tool. Select **Assignments > Pin Planner** to open the display shown in Figure D.1. To make the window look like the one in the figure it may be necessary to enable or disable some of the settings under the **View** menu. The settings enabled in Figure D.1 are **View > Show > Package Top**, **View > Show > Show Fitter Placements**, and **View > All Pins List**.

The image at the top of Figure D.1 depicts the chip package for the EP2C35F672C6 device as viewed from the top. Although a lot of information is available in this window, it is not necessary to examine these details for the purpose of making pin assignments. The locations of pins are identified by row and column indices, where rows are specified using

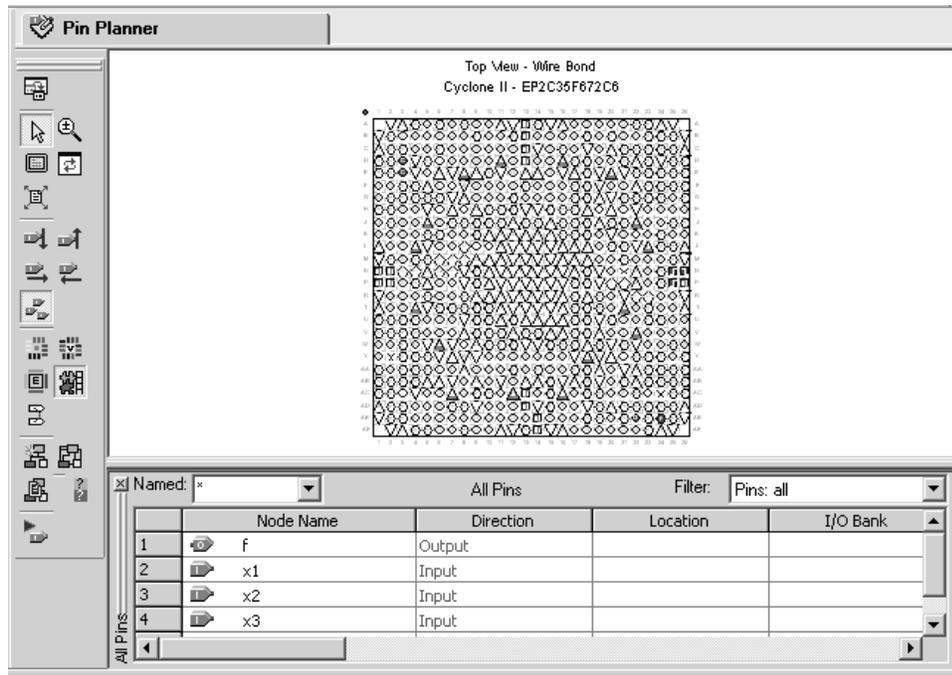


Figure D.1 The Pin Planner display.

letters and columns are specified using numbers. For example, the pin in the fifth column of the top row is called pin A5, and the pin in the fifth column of the bottom row is called pin AF5. The pins that are actually used for a compiled circuit are filled in with a solid color. It is possible to hover the mouse cursor over a pin symbol to open a Tooltip that shows the name of the signal assigned to the pin (if Tooltips are not enabled, select **Tools > Options** and then modify the Tooltip settings for the Pin Planner). A legend that describes the various pin symbols can be opened by selecting **View > Pin Legend Window**.

For this tutorial we will assume that the *example_verilog* circuit will be implemented on the DE2 Development and Education board, which is an FPGA-based board available from Altera. A picture of the DE2 board is given in Figure D.2. While this powerful board includes many features, our simple design will use only some of the switches and lights on the bottom edge of the board. The inputs to the circuit, x_1 , x_2 , and x_3 will be assigned to toggle switches called $SW[0]$, $SW[1]$, and $SW[2]$. These switches are connected to the FPGA pins N25, N26, and P25, respectively. The output, f , of our circuit will be connected to the green light called $LEDG[0]$, which is connected to pin AE22.

The table in the bottom of Figure D.1 lists the input and output ports of our design project, and allows these ports to be assigned to specific pins. To make the desired connection for input x_1 , double-click on its **Location** column, as indicated in Figure D.3, and choose pin N25 from the displayed list. Repeat this procedure to complete all of the pin

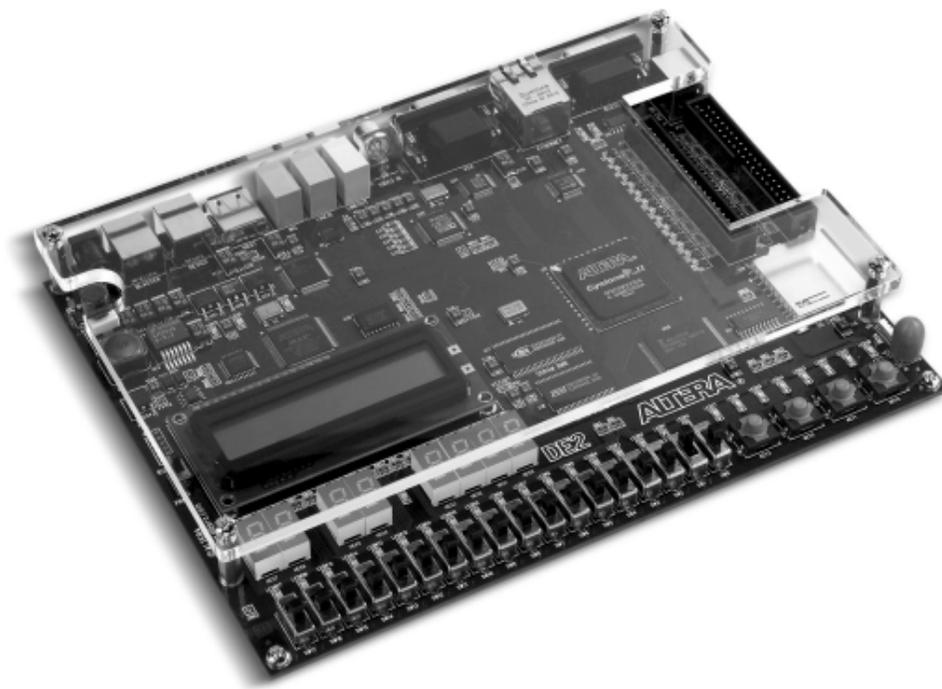


Figure D.2 The Altera DE2 Development and Education Board.

Named: *		All Pins		Filter: Pins: all
	Node Name	Direction	Location	I/O Bank
1	f	Output		
2	x1	Input	PIN_N25	
3	x2	Input		
4	x3	Input		

Figure D.3 Making a pin assignment.

assignments, which leads to the display in Figure D.4. In addition to its use for making new pin assignments, the Pin Planner can also be used to edit or delete existing assignments. A pin assignment can be deleted by selecting it and pressing the **Delete** key on the keyboard.

D.1.1 RECOMPILING THE PROJECT WITH PIN ASSIGNMENTS

Since we have not recompiled the *example_verilog* project, the compilation results have not yet been affected by our pin assignments. To cause the pin assignments to be applied,

The screenshot shows the Pin Planner window with a table of assignments. The window title is 'All Pins' and the filter is set to 'Pins: all'. The table has five columns: Node Name, Direction, Location, and I/O Bank. There are four rows of data, each with a small icon to the left of the node name.

	Node Name	Direction	Location	I/O Bank
1	f	Output	PIN_AE22	7
2	x1	Input	PIN_N25	5
3	x2	Input	PIN_N26	5
4	x3	Input	PIN_P25	6

Figure D.4 The completed pin assignments.

recompile the project. During the compilation process the Fitter uses the pin assignments for the ports that have been specified manually and makes automatic pin assignments for other ports (some special ports that are used for programming and configuring the FPGA are automatically created during the compilation process, and can be seen in the Pin Planner). The Pin Planner tool should now display the correct pin assignments.

D.2 PROGRAMMING AND CONFIGURING THE FPGA DEVICE

Once the circuit has been compiled, it can be downloaded into the FPGA chip on the DE2 board. The board supports a programming mode known as *JTAG* programming. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board. To use this connection, it is necessary to have the USB-Blaster driver installed. If this driver is not already installed, consult the tutorial *Getting Started with Altera's DE2 Board*, which is available on Altera's web site, 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. A reader who does not have access to a DE2 board will not be able to download the circuit, but the steps involved are still easy to follow.

D.2.1 JTAG PROGRAMMING

The programming and configuration task is performed as follows. Make sure that the RUN/PROG switch on the DE2 board is set to the RUN position. Select **Tools > Programmer** to reach the window in Figure D.5. 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

D.2 PROGRAMMING AND CONFIGURING THE FPGA DEVICE

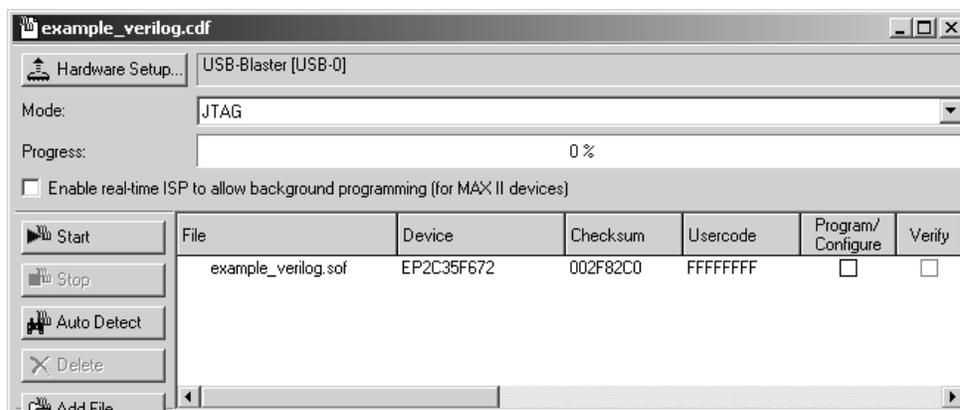


Figure D.5 The Programmer window.

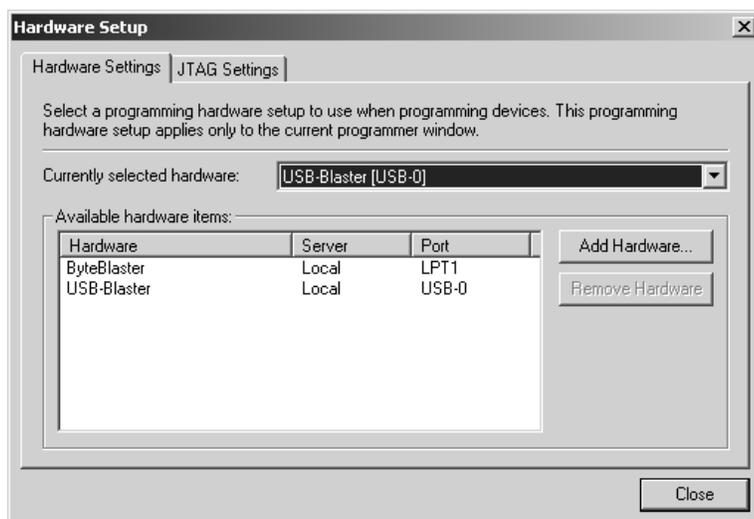


Figure D.6 The Hardware Setup window.

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 D.6.

Now, press Start in the window in Figure D.7. A blue LED on the board will light up when the configuration data has been downloaded successfully. If you see an error reported by the Quartus II software indicating that programming failed, then check to ensure that the board is properly powered on.

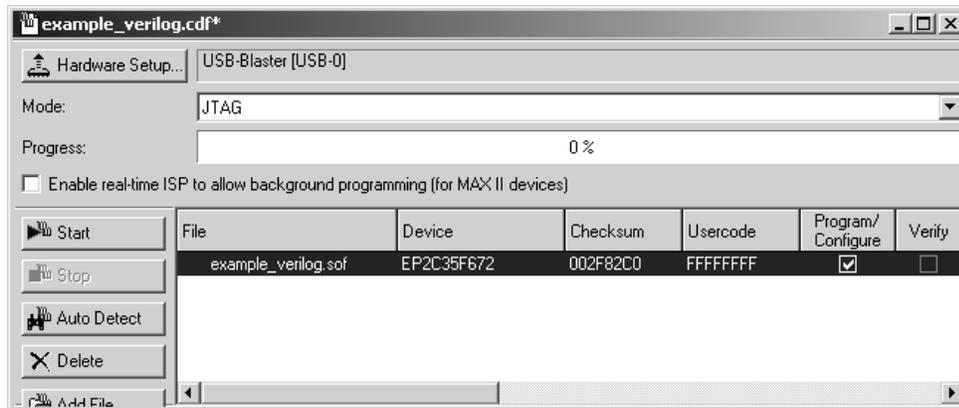


Figure D.7 The updated Programmer window.

Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. Try all eight valuations of the input variables x_1 , x_2 , and x_3 , by setting the corresponding states of the switches $SW[0]$, $SW[1]$, and $SW[2]$. Verify that the circuit properly implements the logic function specified in the *example_verilog* code. If the circuit does not appear to work properly, make sure that you have entered and compiled the correct pin assignments. 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, recompile the circuit, and program the board as explained above.

In our *example_verilog* code we used the input and output port names x_1 , x_2 , x_3 , and f . Another choice would be to use port names that correspond to the names of the switches and lights that are being used on the DE2 board. This may be convenient because the board includes a label adjacent to each switch and light, which makes for easy identification of the circuit's inputs and outputs.

As a simple exercise, modify the *example_verilog* code as illustrated in Figure D.8. In this modified code we have used the port names that correspond to the DE2 board, and we have also added additional outputs called $LEDR[0]$, $LEDR[1]$, and $LEDR[2]$. These outputs correspond to the red lights that appear directly above the toggle switches on the board; they are connected to the FPGA pins AB21, AF23, and AE23, respectively. Assigning the value of a switch to the corresponding light, as done in the Verilog code, causes the light to be illuminated when the switch is in the up (1) position and to be off when the switch is in the down (0) position.

Before compiling the modified code, we need to remove the old pin assignments created for the ports x_1 , x_2 , x_3 , and f , as described in section D.1. Next, we need to create new pin assignments for the ports $SW[0]$, $SW[1]$, $SW[2]$, $LEDR[0]$, $LEDR[1]$, $LEDR[2]$, and $LEDG[0]$. These pin assignments can be entered into the Quartus II software manually by using the Pin Planner, or they can be entered by importing a special file provided by Altera, as described below.

For convenience, especially when using large designs, all relevant pin assignments for the DE2 board are given in a file called *DE2_pin_assignments.csv*. This file includes pin

```
module example verilog (SW, LEDR, LEDG);
  input  [2:0] SW;
  output [2:0] LEDR;
  output [0:0] LEDG;
  wire   x1, x2, x3, f;

  assign x1 = SW[0];
  assign x2 = SW[1];
  assign x3 = SW[2];
  assign LEDR = SW;

  assign f = (x1 & x2) | (~x2 & x3);
  assign LEDG[0] = f;

endmodule
```

Figure D.8 Using port names from the DE2 board.

assignments for all of the port names that appear in the *DE2 User Manual*, which includes the signals *SW[0]*, *SW[1]*, and so on. The file is stored in a simple format called the comma separated value (CSV) format, which is a plain ASCII text file. The file can be found on Altera's DE2 web pages.

The pin assignments in the *DE2_pin_assignments.csv* file can be added to a Quartus II project by using the command **Assignments > Import Assignments**, and then browsing to select the file. Since the signals *SW*, *LEDR*, and *LEDG* are specified in the *DE2_pin_assignments.csv* file as elements of vectors, we must refer to them in the same way in the Verilog design file, as we have done in Figure D.8. Once the pin assignments have been imported, they can be viewed in the Pin Planner window. We should note that the *DE2_pin_assignments.csv* file includes pin assignments for many ports that are not used in our small circuit; extra pin assignments that are imported into a project can be safely ignored.

Make the required pin assignments for the modified Verilog code, either by creating them manually or by importing the *DE2_pin_assignments.csv* file. Recompile the Quartus II project, and then download and test the resulting circuit on the DE2 board.

D.3 CONCLUDING REMARKS

In Tutorials 1, 2, and 3, we have introduced many of the most important features of the Quartus II software. However, many other features are available. The reader can learn about the more advanced capabilities of the CAD system by exploring the various commands and on-line help provided in each application. An extensive set of tutorials for Quartus II can be found on the University Program section of Altera's web site.

