Appendix C

Tutorial 2 — Implementing Circuits in Altera Devices

In this tutorial we describe how to use the physical design tools in Quartus II. In addition to the modules used in Tutorial 1, the following Quartus II modules are introduced: Fitter, Floorplan Editor, and Programmer. To illustrate the procedures involved, we will first implement the example_verilog project created in Tutorial 1 in a MAX 7000 CPLD.

C.1 Implementing a Circuit in a MAX 7000 CPLD

Select File | Open Project and browse to the directory that contains the Verilog design example used in Tutorial 1. As depicted in Figure C.1, select the example_verilog project and click Open.

![Figure C.1. Opening the example_verilog project.](image)

C.1.1 Selecting a Chip

In Tutorial 1 we used the Compiler to perform the initial synthesis that generated the information needed for functional simulation. In this tutorial we will implement the design in a CPLD and then use timing simulation.
Before starting the Compiler, it is necessary to specify which chip to use. Select Assignments | Device to open the window shown in Figure C.2. To select the MAX 7000 CPLD family, click on the pull-down menu in the box labeled Family: and select MAX7000S. The S at the end of the name refers to the members of the MAX 7000 family that are in-system programmable. Methods of CPLD programming are discussed in section 3.6.4. Note that in some cases Quartus II will display the message “Device family selection has changed. Do you want to remove all pin assignments?” Click Yes to close this pop-up.

![Figure C.2. Selecting a MAX7000S device.](image)

In the Target device box you can specify that Quartus II should automatically select a device during compilation. The ability to have a chip chosen automatically, based on the complexity of the circuit that has to be implemented, is sometimes convenient for the designer. In this case we wish to select a specific chip, so click on the setting Specific device selected in 'Available devices' list.

The available chips in the MAX 7000S family are displayed in the box labeled Available devices. One available chip is the EPM7128SLC84-7. The meaning of the chip name is as follows: The EPM7 means that the chip is a member of the MAX 7000 family, and the 128 gives the number of macrocells in the chip. The designator LC84 indicates an 84-pin PLCC package; this type of package is described in section 3.6.3. The −7 gives the speed grade. We discuss speed grades in Appendix E. As indicated in Figure C.2, click on the EPM7128SLC84-7 device, then click OK to close the Assignments window.

C.1.2 Examining the Implemented Circuit

Compile the example_verilog project by selecting Processing | Start Compilation, or by using the Toolbar icon that looks like a solid purple triangle. As we saw in Tutorial 1, the compilation progress through each
Quartus II module is displayed in the Status window on the left side of the Quartus II display. The Analysis & Synthesis module converts the Verilog code into a circuit that comprises macrocells, and the Fitter module chooses locations on the device for these macrocells.

When compilation is finished, the compilation report displayed in Figure C.3 is produced. As we said in Tutorial 1, there is a lot of useful information in this report. As displayed in Figure C.4, click on the small + symbol to expand the Fitter section of the report, and then click on the Fitter Equations section. Scroll through this part of the report to see the logic expressions implemented in our circuit. At the bottom of the report the following expression is given for the output \( f \)

\[
f = \text{OUTPUT}(\text{A1L6});
\]

This expression means that \( f \) appears on an output pin, and that output is defined by the logic expression called A1L6. This logic expression is given near the top of the Fitter Equations section, as displayed in Figure C.4. It is easy to see that these expressions properly implement our logic function \( f = x_1 x_2 + \overline{x_2} x_3 \).

![Figure C.3. The compilation summary.](image)

![Figure C.4. The Fitter Equations section.](image)

### C.1.3 Performing Timing Simulation

Timing simulation is done by using the same procedure that we described in Tutorial 1 for functional simulation. Select Assignments | Settings and click on the Simulator Mode item, as shown in Figure B.25. Open the drop-down list next to Simulation mode: and change this setting from Functional to Timing.
The input waveforms for $x_1$, $x_2$, and $x_3$ that were drawn with the Waveform Editor in Tutorial 1 can be used as inputs for the timing simulation. Select Processing | Start Simulation to run the simulation. When the simulation is completed, the simulation report is displayed. Part of this report is shown in Figure C.5. Select View | Fit in Window to see the complete time range of the waveforms. Compare these waveforms to those shown in Figure B.26. The timing simulation produces the same results as the functional simulation in Tutorial 1 except that the times at which changes in $f$ occur are now determined by the timing characteristics of the EPM7128SLC84-7 chip.

On the simulation report waveforms, we can use the vertical reference line to determine the exact time when $f$ changes value. To do this you may have to select View | Snap to Transition, so that your mouse pointer will align perfectly with an edge on any waveform. Click and drag the vertical reference line, as shown in the figure, to the point where $f$ first changes to 1. The box labeled Master Time Bar: now displays 27.5 ns, meaning that it takes 7.5 ns for the change in $x_3$, which occurs at 20 ns, to cause a change in $f$. This result is a reflection of the $-7$ speed grade of the chip, which is specified as having a delay from an input to an output pin of 7.5 ns.

![Figure C.5. The Timing Simulation Report.](image)

### C.1.4 Using the Floorplan Editor

In section C.1.2 we showed that the implementation results produced by the Compiler can be examined by looking at the equations in the compilation report. An alternative to view the implementation results is to use the Floorplan Editor. Select Assignments | Last Compilation Floorplan to open the window shown in Figure C.6. Another way to open this window is to click on Floorplan view in the Fitter section of the compilation report. To make the window look like the one in the figure, it may be necessary to change some of the settings in the Floorplan tool. Selecting View | Interior Cells causes the macrocells in the device to be displayed, and choosing View | Full Screen causes the Floorplan tool to be expanded to fill the screen. Figure C.6 shows some of the macrocells in the EPM7128SLC84-7 chip. As we describe in Appendix E, the macrocells are organized into logic array blocks (LABs), where each LAB contains 16 macrocells. To see larger or smaller views of the LABs, click on the magnify buttons on the left edge of the window. To display different sections of the chip, use the window scroll bars.
The Floorplan Editor uses different colors to indicate macrocells that are used in a circuit and macrocells that are unused. For our small example three input pins are used for the three inputs to the circuit, and one macrocell provides the circuit output. Orient the display so that the macrocell that produces the output \( f \) is visible, as depicted in Figure C.7. Click on this macrocell to select it. The Floorplan Editor can draw lines that indicate which other macrocells the selected macrocell is connected to by selecting View | Routing | Show Node Fan-In. It is also possible to see what logic function is implemented in the selected node by selecting View | Equations. As illustrated in the figure, this choice displays the logic equation from the compilation report in the bottom part of the Floorplan window.

Instead of displaying the macrocells, the Floorplan tool can alternatively display a picture of the package pins on the target chip. To change to this view, select View | Package Top. To close the report file equation viewer, select again View | Equations to toggle off this feature. It is possible to make visible the entire device by clicking on the “negative” magnify button a few times, or by selecting View | Fit in Window (shortcut Ctrl-w). The “positive” magnify button and scroll bars can be used to change the view until the signal names assigned to the pins are large enough to read. Figure C.8 gives an example of the display oriented so that the pins used for input \( x3 \) and output \( f \) are visible. The scroll bars can be used to examine the rest of the pins.
Figure C.7. Viewing node fan-in and equations.

Figure C.8. The package top view.
The Floorplan tool is not essential in the CAD flow described above. It just provides a graphical view of the information contained in the compilation report. We will describe a different use of the Floorplan tool in section C.4.3, in which it will be used to modify the implementation results produced by the Compiler, instead of just displaying them.

We have now completed the implementation of the example_verilog project in a MAX 7000 chip. To close this project select File | Close Project.

C.2 Implementing a Circuit in a FLEX 10K FPGA

The CAD flow used to implement a circuit in a FLEX 10K FPGA is the same as that used for the MAX 7000 CPLD. We showed in Chapter 4 that multilevel logic synthesis is an effective optimization strategy when targeting designs to lookup table–based FPGAs. Figure 4.54 gives Verilog code for a seven-variable logic function used to illustrate the benefits of multilevel synthesis. In this section we will create a new design project, named example_verilog2, which represents the Verilog code in that figure.

To create a new project select File | New Project Wizard. In Figure C.9 we specify the directory name tutorial2 and use the name example_verilog2 for both the project name and the name of the top-level entity. Click Next to proceed to the second screen of the New Project Wizard, which allows design files to be added to the project. If Quartus II prompts whether it should create the directory for the new project, select Yes.

Since we do not yet have any design files to add to the project, click Next to move to the screen that allows EDA tools to be specified. We are not using any EDA tools, so choose Next again to proceed to the device selection screen, as shown in Figure C.10. Open the Family: drop-down box and select FLEX10K. Allow Quartus II to automatically choose an appropriate device for the project by clicking on the setting No, I want to allow the Compiler to choose a device. Click Finish to complete the New Project Wizard.
To create the Verilog design file, select File | New, which opens the dialogue in Figure B.11. Choose Verilog HDL File as the file type and click OK. To give the file the proper name select File | Save As and specify the filename example_verilog2. As discussed for Figure B.27 make sure the setting Add file to current project in the Save As dialogue is checked. Type the code from Figure 4.54 into the Text Editor, as displayed in Figure C.11a.

Compile the project in the FLEX10K FPGA, and fix any errors reported. After successful compilation, in the compilation report expand the Fitter section and click on Fitter Equations. At the bottom of this section in the report, the output $f$ is specified as

$$f = \text{OUTPUT}(A1L3);$$

As shown in Figure C.11b the logic expression for A1L3 implements $f$ in a multi-level logic form. The first level of logic is specified as

$$A1L2 = x_7(x_2 + x_1\overline{x_6}) + \overline{x_7}(x_1\overline{x_6})$$

We show in Appendix E that the logic cell in the FLEX 10K FPGA is a 4-input lookup table (LUT) that can implement any four-input function. Since the expression above has four inputs, it can be realized in one logic cell in the device. This cell provides an input to the next-level expression

$$A1L3 = A1L2(x_3 + x_4x_5)$$

This expression also has four inputs, and can therefore be realized in a single cell. Thus, $f$ is implemented as two logic cells connected in series. The reader is encouraged to verify that the expression for A1L3 properly implements the function specified in Figure C.11a.

Having implemented the design in the FLEX 10K device, the next step is to perform a timing simulation. Since the steps involved are the same as in our previous examples, we will not show them here. The reader should perform the timing simulation to gain a feeling for the timing characteristics of the FLEX 10K device.
Part a. The Verilog source code.


Figure C.11. The example_verilog2 source code and implementation.

C.3 Downloading a Circuit into a Device

Once a circuit has been compiled for a design project, the circuit can be downloaded into the selected device. Downloading involves programming the appropriate switches in the device to implement the desired circuit. To illustrate the steps involved, we will describe how a circuit can be downloaded into a laboratory development board that is available from Altera Corporation. The board is called the UP−1 Education Board and includes both a MAX 7000 CPLD and a FLEX 10K FPGA. The UP−1 board can be obtained by following the instructions in the University Program section of Altera’s Web site at http://www.altera.com.

We will describe how the example_verilog project that we implemented in a MAX 7000 CPLD can be downloaded into the UP−1 board, assuming that it is connected to the reader’s computer. A reader who does not have access to the UP−1 board will not be able to download the circuit, but the steps involved are still easy to follow. The UP−1 board is connected to the computer using a type of cable that is available from Altera. For purposes of this discussion we will assume that a ByteBlaster cable is used, which provides a connection to a parallel port on the computer.

The UP−1 board contains an EPM7128SLC84-7 chip. There is a socket that connects this chip to the ByteBlaster cable. Plug the ByteBlaster cable into this socket and plug the other end of the cable into the parallel port on the computer. Ensure that the UP−1 board is plugged into a power supply and that the green “power LED” is lit.

Use File | Open Project to open the example_verilog project. Select Tools | Programmer to open the Programmer module window shown in Figure C.12. The programming file for the example_verilog project,
which is called example_verilog.pof, should be listed in the Programmer window. If this file is not shown click Edit | Add File and type the file name.

![Figure C.12. The Programmer module window.](image)

To specify that the ByteBlaster is to be used as the programming hardware, browse on the Hardware... button, which opens the window in Figure C.13a. If the ByteBlaster is not listed under Available hardware items: click on the Add Hardware button. This action opens the window in Figure C.13b. Open the drop-down list next to Hardware type: and select the item ByteBlasterMV or ByteBlaster II. Click OK to return to the Hardware Setup window (Figure C.13a).

![Part a. The Hardware Setup window.](image)

![Part b. The Add Hardware dialogue.](image)

Figure C.13. Adding the ByteBlaster hardware.
The ByteBlaster should now appear in the **Available hardware** items: box. Click on this item to highlight it, and then click the **Select Hardware** button. Close the Hardware Setup window to return to the Programmer window in Figure C.14. Notice that the ByteBlaster is now shown to the right of the **Hardware** button, meaning that this cable is now selected. As indicated in the figure, click on the two checkmark boxes under **Program/Configure** and **Verify** associated with the `example_verilog` file.

![Figure C.14. The final Programmer module window.](image)

To configure the EPM7128SLC84-7 chip, select **Processing** | **Start Programming**. The Programmer module automatically downloads the `example_verilog.pof` file through the ByteBlaster cable into the device and then verifies that the programming has been performed correctly. The Programmer module can now be closed. The designer can test the circuit implemented in the chip by using appropriate test equipment.

The UP−1 board also contains a FLEX 10K chip. The procedure used to download a circuit into this chip is similar to the one described for the MAX 7000 device, but a few extra steps are needed. The reader who tries using the FLEX 10K chip should refer to the documentation that accompanies the UP−1 board for detailed instructions.

### C.4 Making Pin Assignments

In the examples given so far in this tutorial, the assignment of signals to device pins is 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 chip(s) being used may have hardwired connections from some of the device 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 signals should be assigned to.

In section C.1.4 we described how to examine the compilation results by using the Last Compilation Floorplan tool. Figure C.8 presented the top view of the chip package and showed the assignments of signals $x_3$ and $f$ to pins 4 and 12, respectively. In section C.4.3 we will show how the pin assignments can be changed by using the Floorplan tool. However, Quartus II provide several ways of making pin assignments, and we will first describe the method that uses the **Assignments** dialogue.

Before pins can be assigned manually, it is necessary to first specify which chip to use. This was already done in section C.1.1, when we selected the EPM7128SLC84-7 as shown in Figure C.2. Select **Assignments** | **Assign Pins** to open the window in Figure C.15. The box labeled **Available Pins & Existing Assignments** shows all of the device package pins and it displays assignments after they have been made. As an example, we will assign the inputs $x_1$, $x_2$, and $x_3$ to pins 9, 10, and 11. As indicated in Figure C.15, scroll down until pin 9 is visible and click on this pin to highlight it. In the **Assignment** box
click on the browse . . . button next to the Pin name: item. This operation opens the Node Finder window, shown in Figure C.16.

![Assign Pins dialogue](image1)

**Figure C.15. The Assign Pins dialogue.**

The Node Finder is a useful utility that allows you to search for various components in a design project. As shown in the figure, use the drop-down list next to the Filter: item to specify that Node Finder should search for pins. Clicking the Start button causes the list of pins shown in the Nodes Found: box to be displayed. Click on the x1 input pin and then click on the > button to move this pin to the Selected Nodes: box. Click OK to return to the Assign Pins dialogue. On the window in Figure C.15 click on the Add button. This action causes x1 to appear in the Name column beside pin 9, and substitutes a Change button in place of the Add button, as depicted in Figure C.17.

![Node Finder window](image2)

**Figure C.16. The Node Finder window.**
Figure C.17. The assignment of input $x_1$ to pin 9.

Following the same procedure, click on pin 10 and assign the $x_2$ input, then click on pin 11 and assign the $x_3$ input. The Available Pins & Existing Assignments box should now show the assignments given in Figure C.18. Click OK to close the Assign Pins window. At this point the Assignments dialogue is the active window in the Quartus II display. Although we did not open this dialogue directly, it was opened implicitly when we selected the Assign Pins command. To complete the pin assignment procedure you have to select OK to close the Assignments window. Clicking Cancel to close the Assignments window would discard the pin assignments.

Figure C.18. The pin assignments for inputs $x_1$, $x_2$, and $x_3$.

Since we have not recompiled the example.vlog project, the compilation results have not yet been affected by our pin assignments. At this point, the assignments are stored by Quartus II in a file with extension .csf, which stands for compiler settings file. Select File | Open to examine the example.vlog.csf file in the Text Editor. Scroll through this file, or use Edit | Find, to locate the pin assignments, which have the form

$$x_1 : \text{LOCATION} = \text{Pin}_9;$$
$$x_2 : \text{LOCATION} = \text{Pin}_{10};$$
$$x_3 : \text{LOCATION} = \text{Pin}_{11};$$

It is possible to modify, add, or delete pin assignments by editing this file, but this is not recommended because it is easy to make an error in the required syntax of the file.
C.4.1 Examining Pin Assignments with the Floorplan Editor

As we mentioned before, it is possible to see the pin assignments that result after compilation by using the Last Compilation Floorplan tool. Quartus II also provides a floorplan tool that can display both the pin assignments produced from the last compilation and the user’s pin assignments that have not yet been compiled. Select Assignments | Timing Closure Floorplan to open the floorplan tool presented in Figure C.19. If you do not have the device package view selected, click on View | Package Top.

Under the View | Assignments menu there are two settings: Show User Assignments, and Show Fitter Assignments. If the first setting is active, then the user’s assignments, such as the pin assignments made in section C.4 are displayed, and if the second setting is active then the last compilation results are displayed. Since the two settings are independent you can choose to see both types of assignments, one at a time, or neither. In Part a of Figure C.19 we have activated only the setting View | Assignments | Show Fitter Assignments, which gives the same picture displayed in Figure C.8. Part b of Figure C.19 uses the setting View | Assignments | Show User Assignments, so that the pin assignments we made for inputs $x_1$, $x_2$ and $x_3$ are shown. Each type of assignment can also be shown in the device view, instead of a package view. Select View | Interior Cells to experiment with this feature.

C.4.2 Recompiling the Project with Pin Assignments

To change the compilation results using our pin assignments, select Processing | Start Compilation, or use the appropriate shortcut to start the Compiler. During the compilation process the Fitter module uses the pin assignments for the signals that have been specified manually and makes automatic pin assignments for the other signals. In the produced compilation report open the Fitter section and click on Floorplan view. Notice that the inputs $x_1$, $x_2$ and $x_3$ now obey the assignments to pins 9, 10, and 11, respectively. Another way to see the compilation results is to select Assignments | Timing Closure Floorplan and activate the setting View | Assignments | Show Fitter Assignments. Since we have now recompiled the design to apply our assignments, the setting View | Assignments | Show User Assignments shows the same pin assignments as in the compilation results.

C.4.3 Changing Pin Assignments by using the Floorplan Editor

We explained at the beginning of section C.4 how to make pin assignments by using the Assignments | Assign Pins dialogue. Another way to create, or change, pin assignments is to use the Floorplan Editor. If not already done, select Assignments | Timing Closure Floorplan and click on View | Package Top. To see both the compilation results and user assignments at the same time, activate both View | Assignments | Show Fitter Assignments and View | Assignments | Show User Assignments. The floorplan display should look similar to the one in Figure C.20. Click on the pin for the output $f$ and drag this pin with the mouse and drop it on pin 15. This operation creates a pin assignment for $f$ that is stored in example_verilog.csf as

\[ f : \text{LOCATION} = \text{Pin}_{15}; \]
Part a. Viewing the Fitter Assignments.

Part b. Viewing the User Assignments.

Figure C.19. Viewing Pin Assignments in the Floorplan Editor.
Any pin assignment can be changed in this manner by using the Floorplan Editor. Another variant for making pin assignments is to open the Node Finder tool to search for pins, and then drag-and-drop the pin names from Node Finder onto package pins in the Floorplan Editor. Of course, new assignments that are created only affects the compilation results when the project is recompiled.

Pin assignments can be deleted from a project by using the same tools employed to create the assignments. In the Assign Pins dialogue, click to highlight an existing pin assignment and then click on the Delete button (see Figure C.17). In the Floorplan Editor, click on a pin that has a pin assignment and select Edit | Delete to remove the assignment. Again, these changes only affect the compilation results when the project is recompiled.

C.5 Concluding Remarks

Having completed this and the preceding tutorial, the reader is familiar with many of the most important features of Quartus II. In the next tutorial we will introduce some additional features that are useful for larger circuits, especially those that contain storage elements.