Chapter 6

Tutorial: Creating an FPGA Project

This tutorial provides comprehensive information for understanding how to create a FPGA design and run it on the DE0-Nano development and education board. The following sections provide a quick overview of the design flow, explaining what is needed to get started, and describe what is taught in this tutorial.

6.1 Design Flow

Figure 6-1shows a block diagram of the FPGA design flow.

The first step in the FPGA design flow starts is design entry. The standard design entry methods are using schematics or a hardware description language (HDL), such as Verilog HDL or VHDL. The design entry step is where the designer creates the digital circuit to be implemented inside the FPGA. The flow then proceeds through compilation, simulation, programming, and verification in the FPGA hardware.

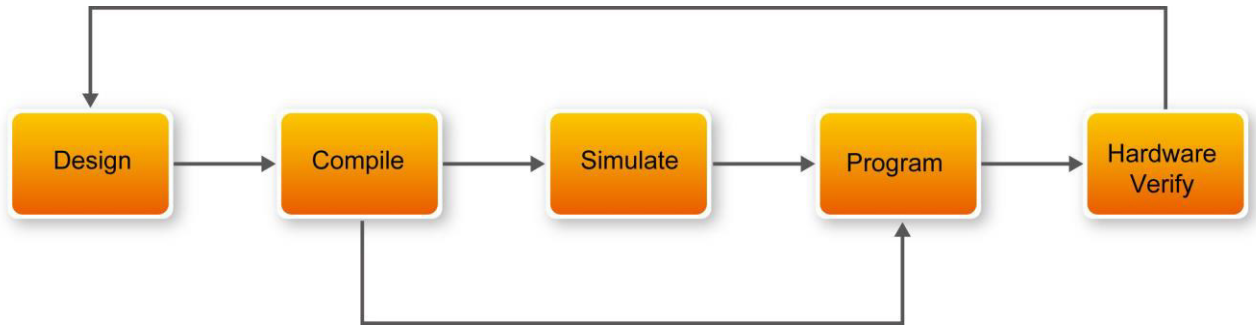

Figure 6-1 Design Flow

This tutorial describes all of the steps except for simulation. Although it is not covered in this document, simulation is very important to learn. There are two types of simulation, Functional and Timing Functional simulation allows you to verify that your hardware is performing the desired functionality. Timing (or post place-and-route) simulation verifies that the design meets timing and functions appropriately in the device. Simulation tutorials can be found on the Altera University Program website at http://university.altera.com.

6.2 Before You Begin

This tutorial assumes the following prerequisites

■ You have a general understanding of FPGAs. This tutorial does not explain the basic concepts of programmable logic.

■ You are somewhat familiar with digital circuit design and electronic design automation (EDA) tools.

■ You have installed the Altera Quartus II 10.1 software on your computer. If you do not have the Quartus II software, you can download it from the Altera web site at www.altera.com/download.

■ You have a DE0-Nano Development Board on which you will test your project. Using a development board helps you to verify whether your design is really working.

■ You have gone through the quick start guide and/or the getting started user guide for your development kit. These documents ensure that you have:

- Installed the required software.
- Determined that the development board functions properly and is connected to your computer.

Next step is to install the USB-Blaster driver, if not already done. To install the driver, connect a USB cable between the DE0-Nano board and a USB port on a computer that is running the Quartus II software.

The computer will recognize the new hardware connected to its USB port, but it will be unable to proceed if it does not have the required driver already installed. If the USB-Blaster driver is not already installed, the New Hardware Wizard in **Figure 6-2** will appear.

Figure 6-2 Found New Hardware Wizard

The desired driver is not available on the Windows Update Web site, therefore select "No, not this time" and click **Next**. This leads to the window in Figure 6-3.

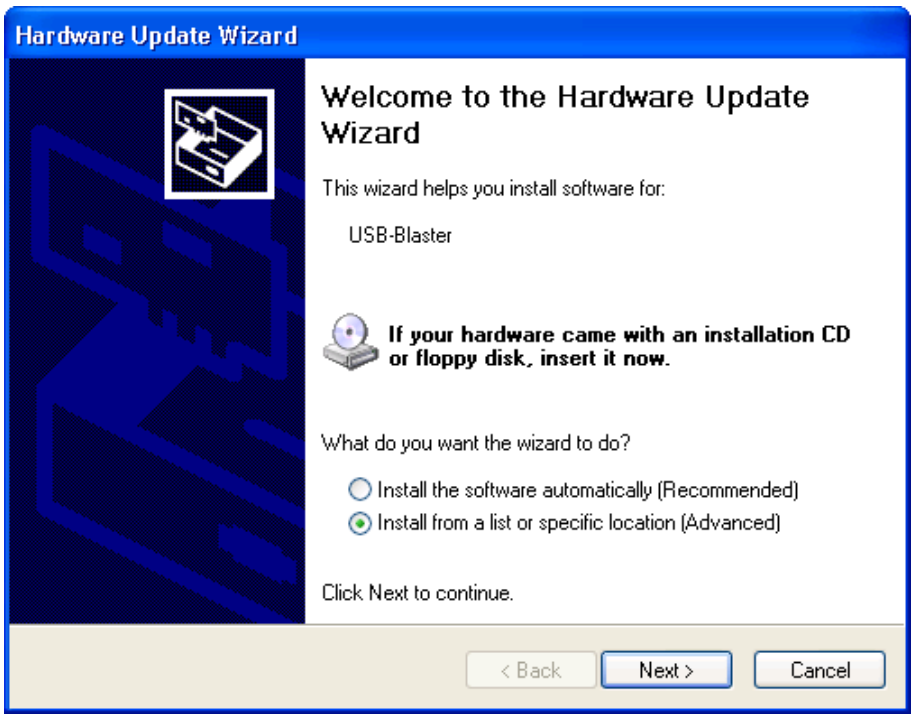

Figure 6-3 The driver is found in a specific location

The driver is available within the Quartus II software. Hence, select "Install from a list or specific location" and click **Next** to get to Figure 6-4.

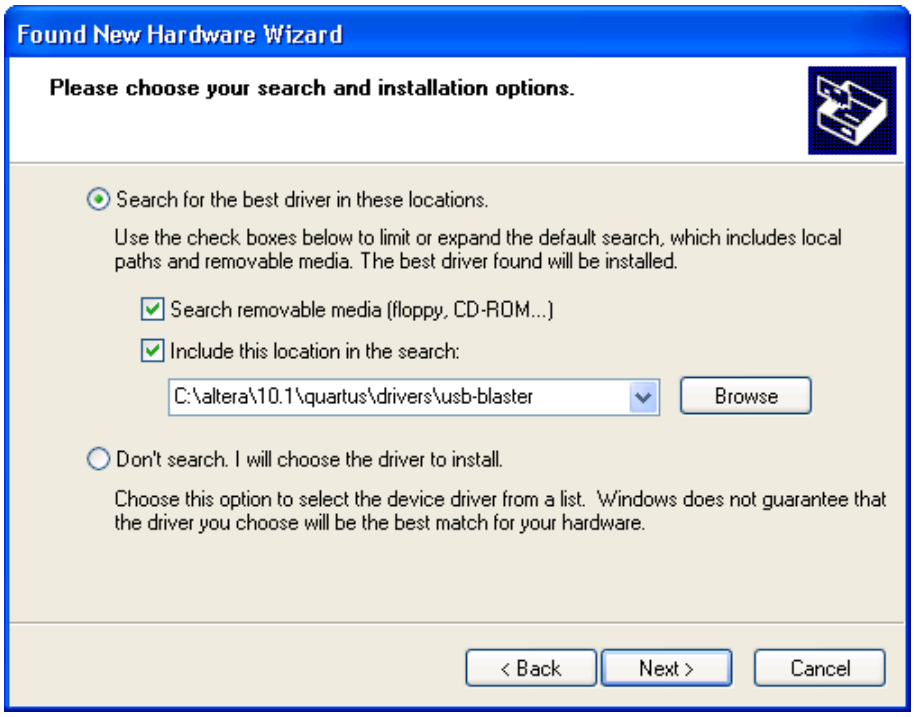

Figure 6-4 Specify the location of the driver

Now, select "Search for the best driver in these locations" and click Browse to get to the pop-up dialog box in Figure 6-5 Find the desired driver, which is at location C:\altera\10.1\quartus\drivers\usb-blaster. Click OK and then upon returning to Figure 6-4 click Next. At this point the installation will commence, but a dialog box in Figure 6-6 will appear indicating that the driver has not passed the Windows Logo testing. Click Continue Anyway.

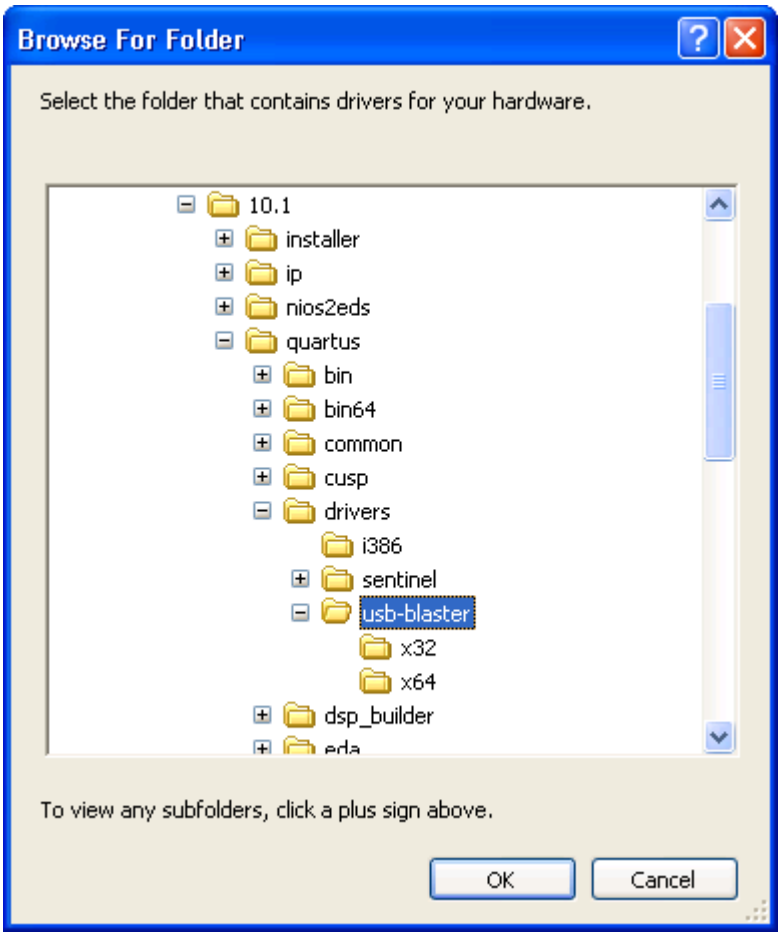

Figure 6-5 Browse to find the location

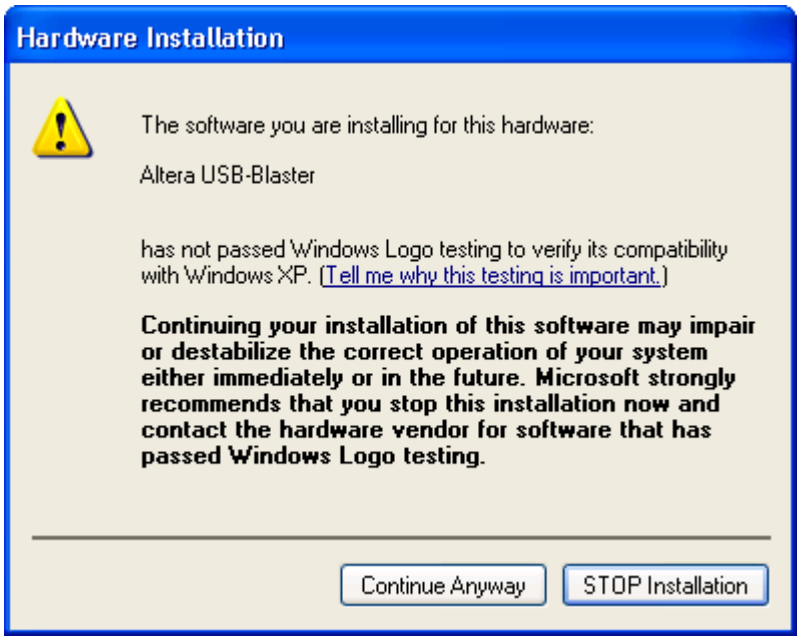

Figure 6-6 There is no need to test the driver

The driver will now be installed as indicated in Figure 6-7. Click **Finish** and you can start using the DE0-Nano board.

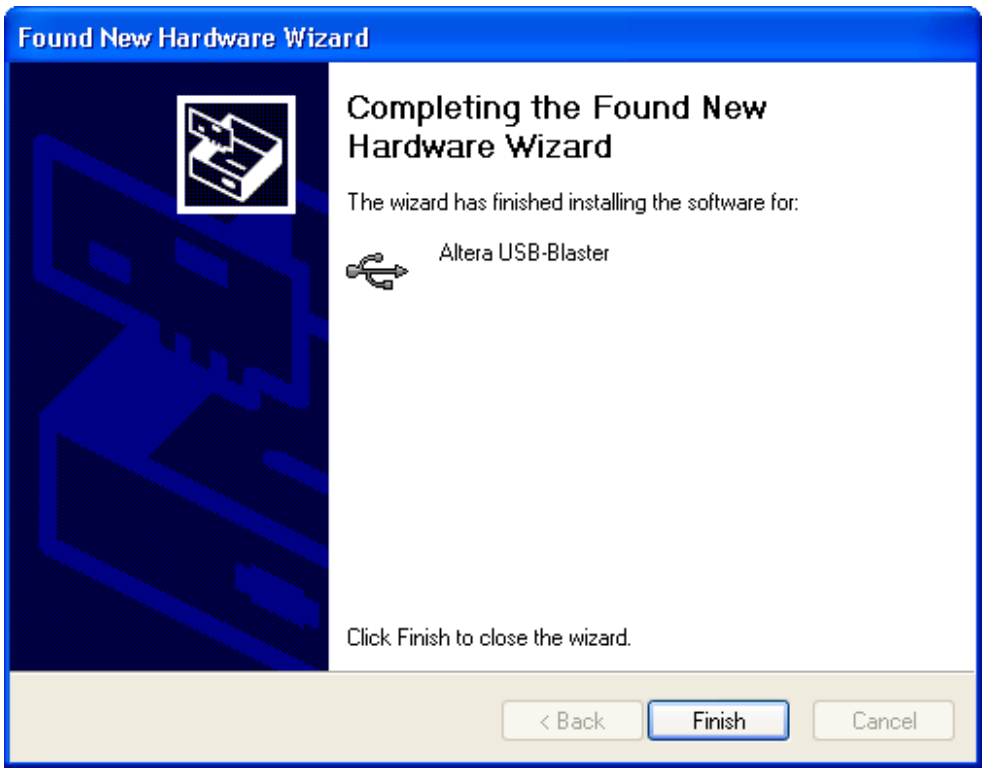

Figure 6-7 The driver is installed

6.3 What You Will Learn

In this tutorial you will perform the following tasks:

Create a design that causes LEDs on the development board to blink at two distinct rates. This design is easy to create and gives you visual feedback that the design works. Of course, you can use your DE0-Nano board to run other designs as well. For the LED design, you will write Verilog HDL code for a simple 32-bit counter, add a phase-locked loop (PLL) megafunction as the clock source, and add a 2-input multiplexer megafunction. When the design is running on the board, you can press an input switch to multiplex the counter bits that drive the output LEDs.

6.4 Assign The Device

Begin this tutorial by creating a new Quartus II project. A project is a set of files that maintain information about your FPGA design. The Quartus II Settings File (.qsf) and Quartus II Project File (.qpf) files are the primary files in a Quartus II project. To compile a design or make pin assignments, you must first create a project. The steps used to create a project are:

1. In the Quartus II software, select **File > New Project Wizard**. The Introduction page opens, as shown in **Figure 6-8.**

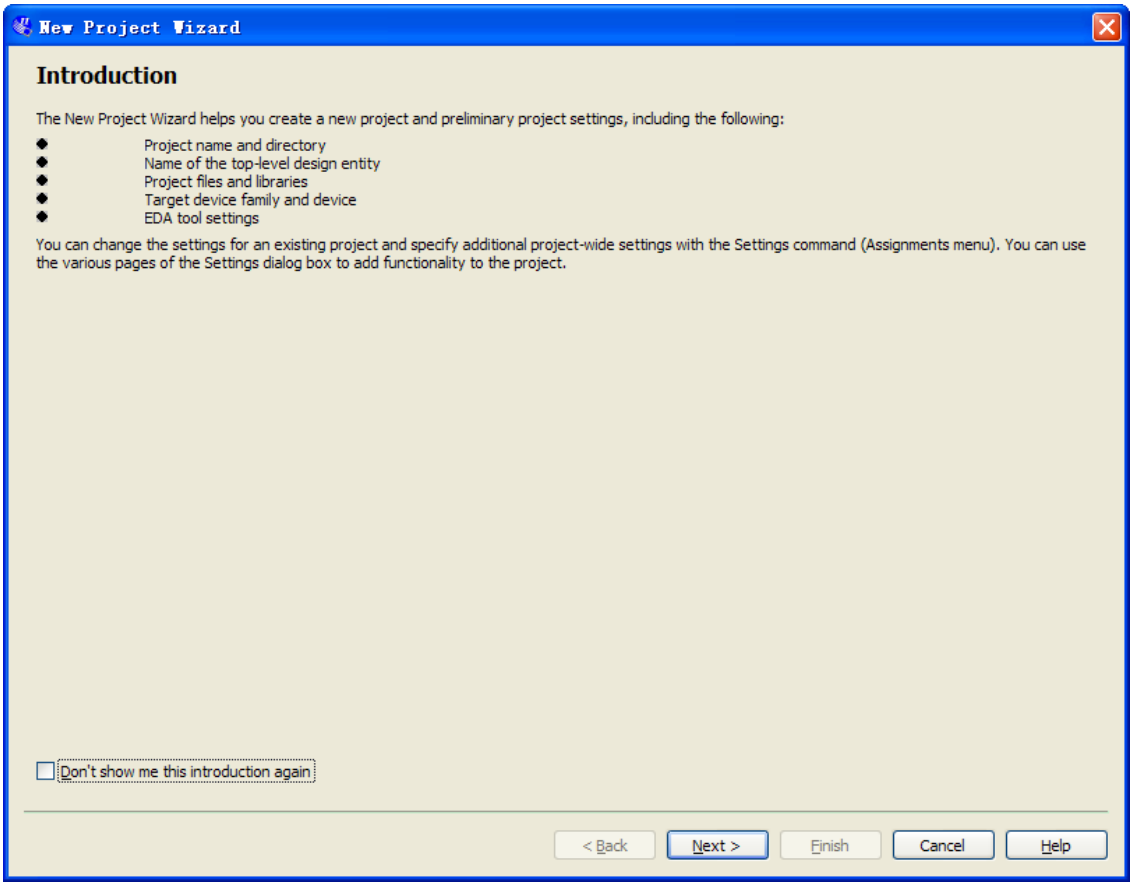

Figure 6-8 New Project Wizard introduction

2. Click **Next**.

3. Enter the following information about your project: (Note: File names, project names, and directories in the Quartus II software cannot contain spaces.)

a. What is the working directory for this project? Enter a directory in which you will store your Quartus II project files for this design. For example, **E:\My_design\my_first_fpga**.

b. What is the name of this project? Type **my_first_fpga**.

c. What is the name of the top-level design entity for this project? Type **my_first_fpga**. See **Figure 6-9**.

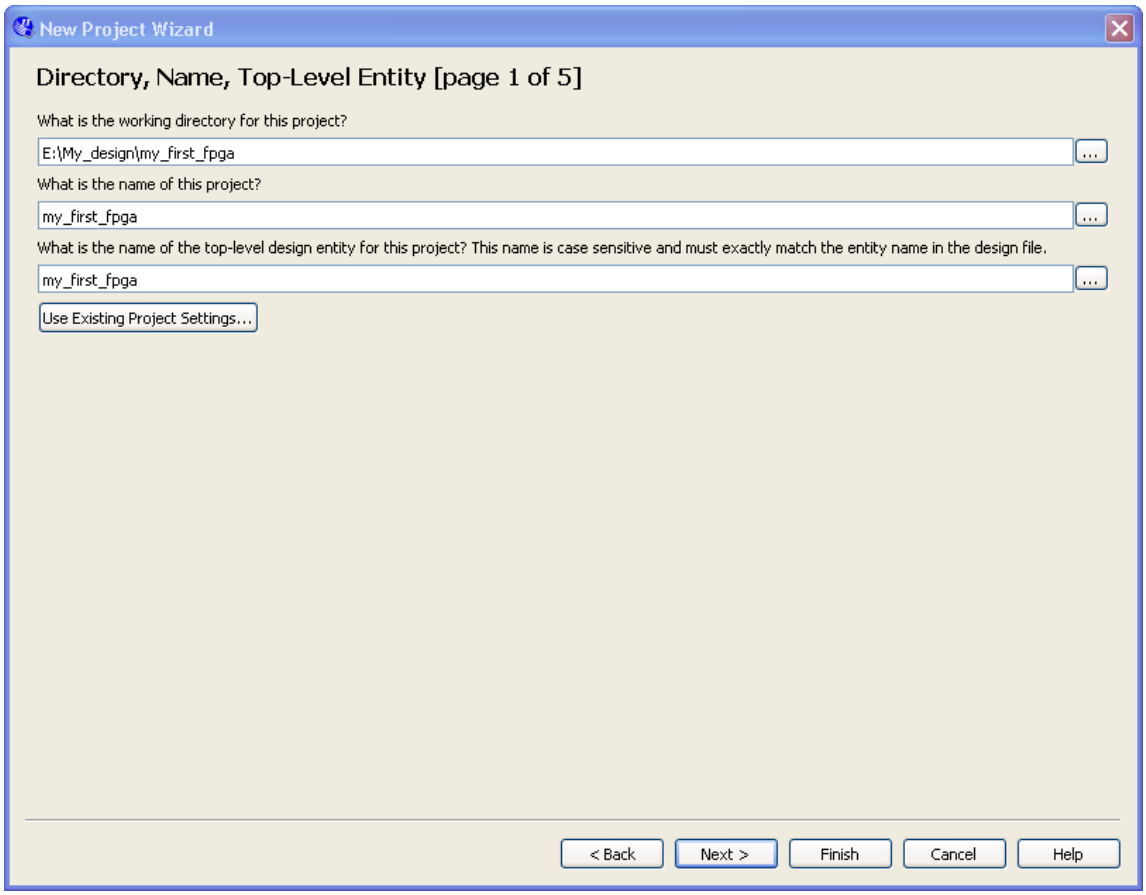

Figure 6-9 Project information

d. Click **Next**.

e. In the next dialog box, you will assign a specific FPGA device to the design. Select the **EP4CE22F17C6** device, as it is the FPGA on the DE0-Nano, as shown in Figure 6-10.

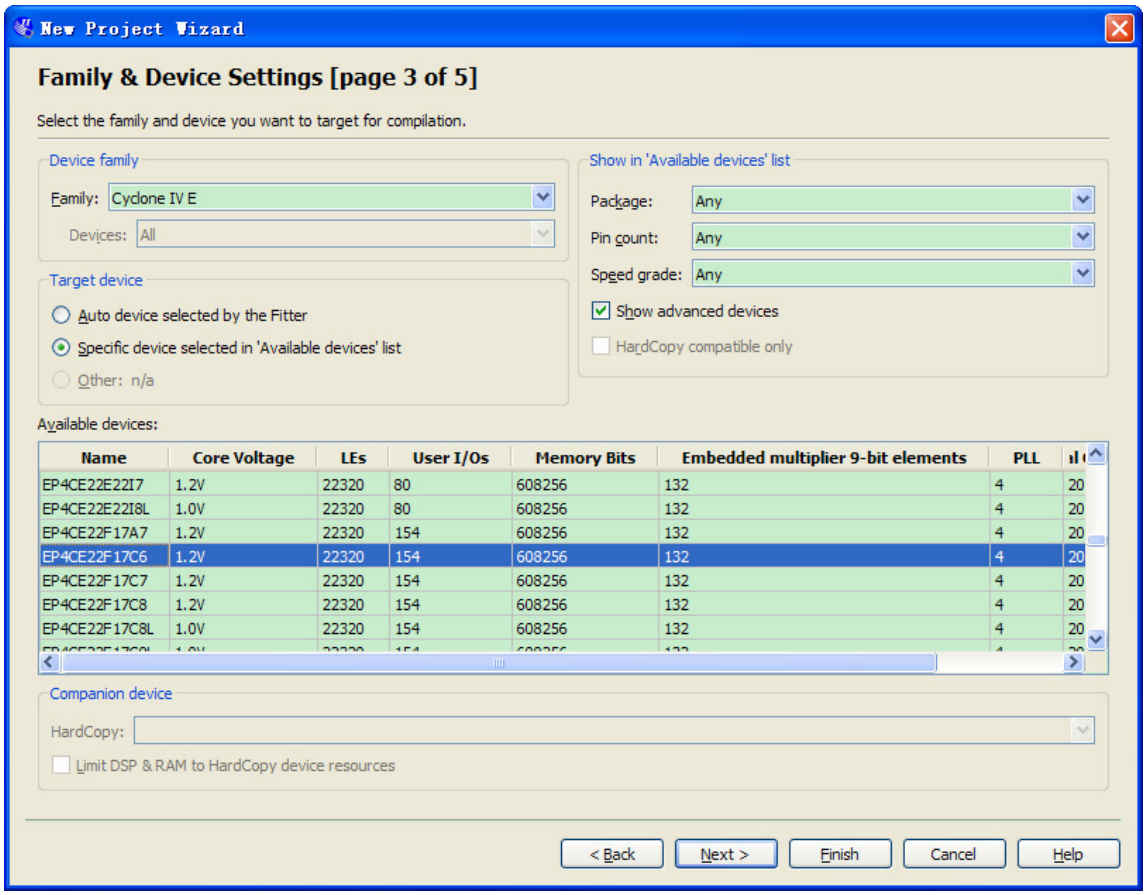

Figure 6-10 Specify the Device Example

f. Click **Finish**.

4. When prompted, select **Yes** to create the my_first_fpga project directory. You just created your Quartus II FPGA project. Your project is now open in Quartus II, as shown in Figure 6-11.

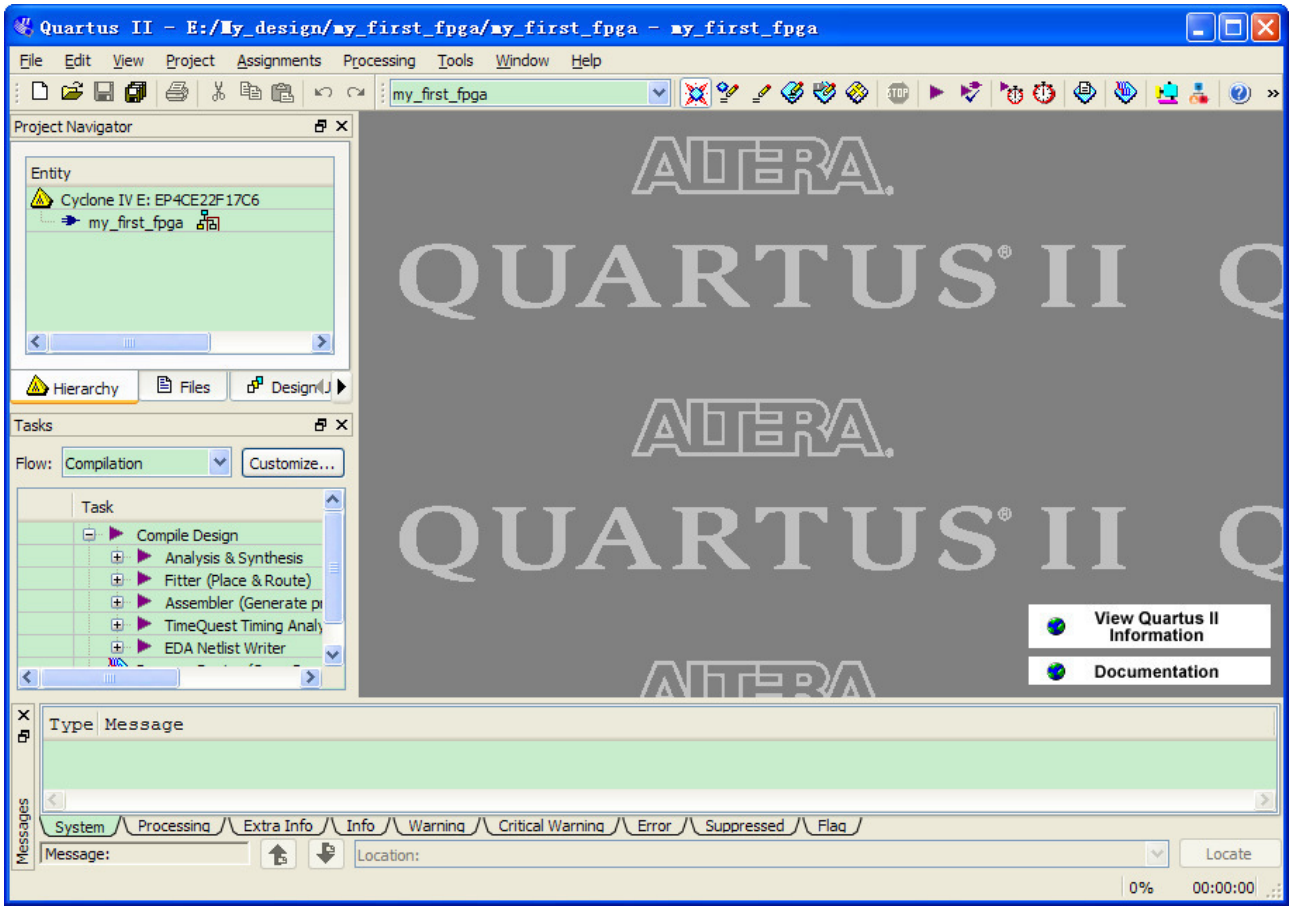

Figure 6-11 my_first_fpga project

6.5 Creating an FPGA design

This section describes how to create an FPGA design. This includes creating the top-level design, adding components (in Verilog HDL and using the megafunctions), adding pins and interconnecting all the components and pins.

First, create a top-level module. In this tutorial, you will use schematic entry, via a Block Design File (.bdf). Alternatively, you could use Verilog HDL or VHDL for the top-level module. The following steps describe how to create the top-level schematic.

1. Select File > New > Block Diagram/Schematic File (see **Figure 6-12** to create a new file, Block1.bdf, which you will save as the top-level design.

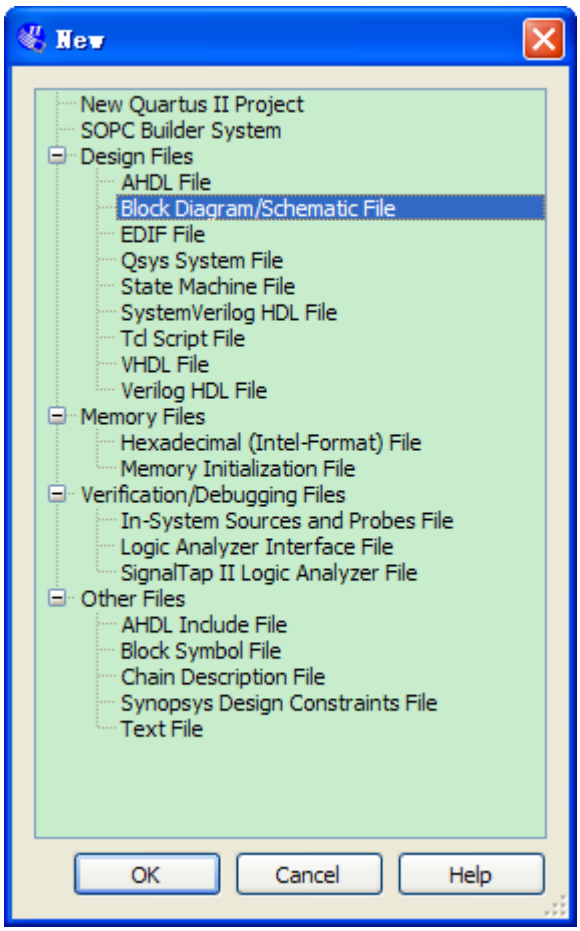

Figure 6-12 New BDF

2. Click **OK**.
3. Select **File > Save As** and enter the following information.

- File name: my_first_fpga
- Save as type: Block Diagram/Schematic File (*.bdf)

4. Click **Save**. The new design file appears in the Block Editor (see **Figure 6-13**).

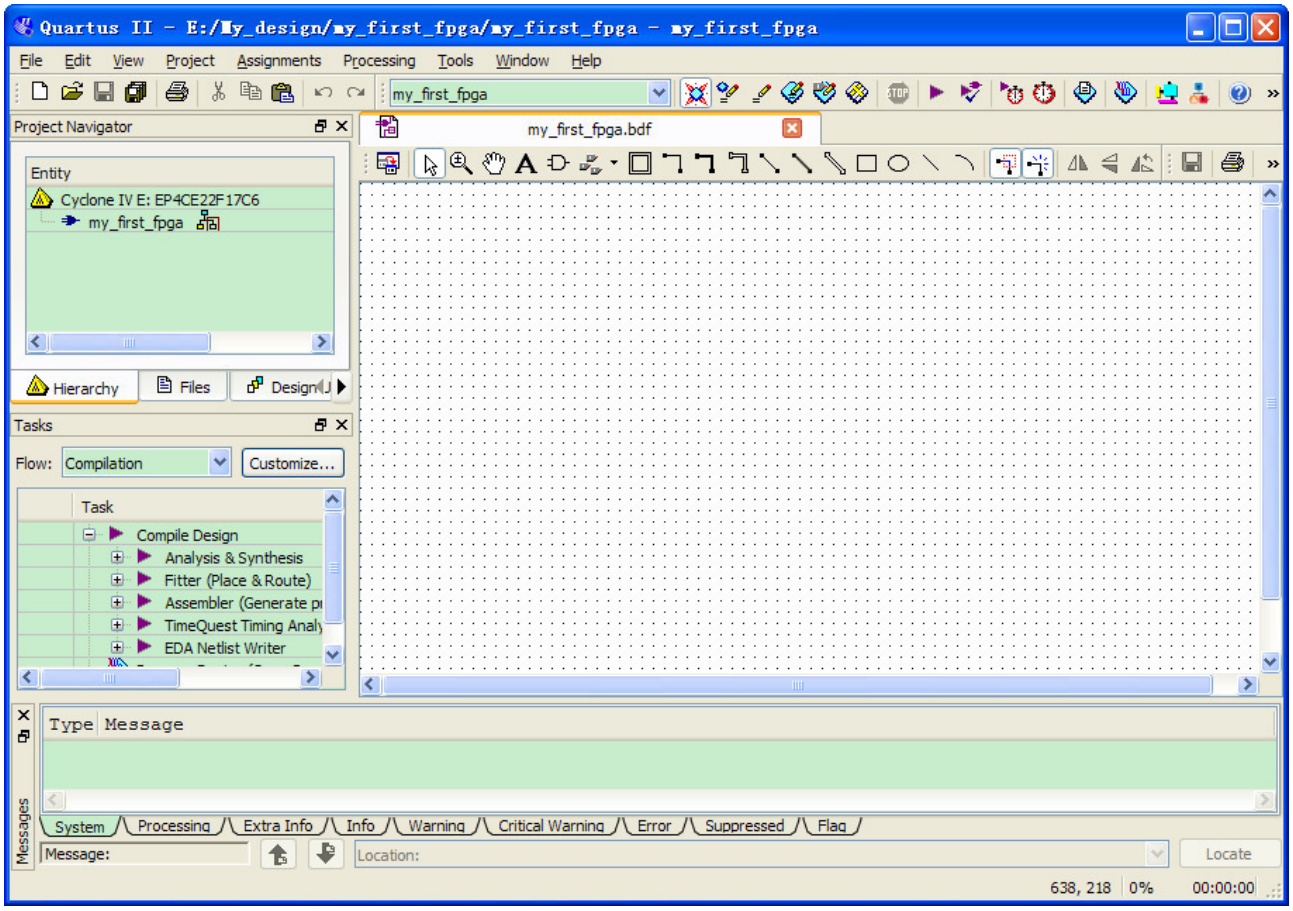

Figure 6-13 Bank BDF

■ Adding a Verilog HDL to the Schematic

1. Add HDL code to the blank block diagram by choosing File > New > Verilog HDL File.
2. Select **Verilog HDL File** in the tree and Click **OK**.
3. Save the newly created file, by selecting **File > Save As** and entering the following information (see Figure 6-14).

- File name: simple_counter.v
- Save as type: Verilog HDL File (*.v, *.vlg, *.verilog)

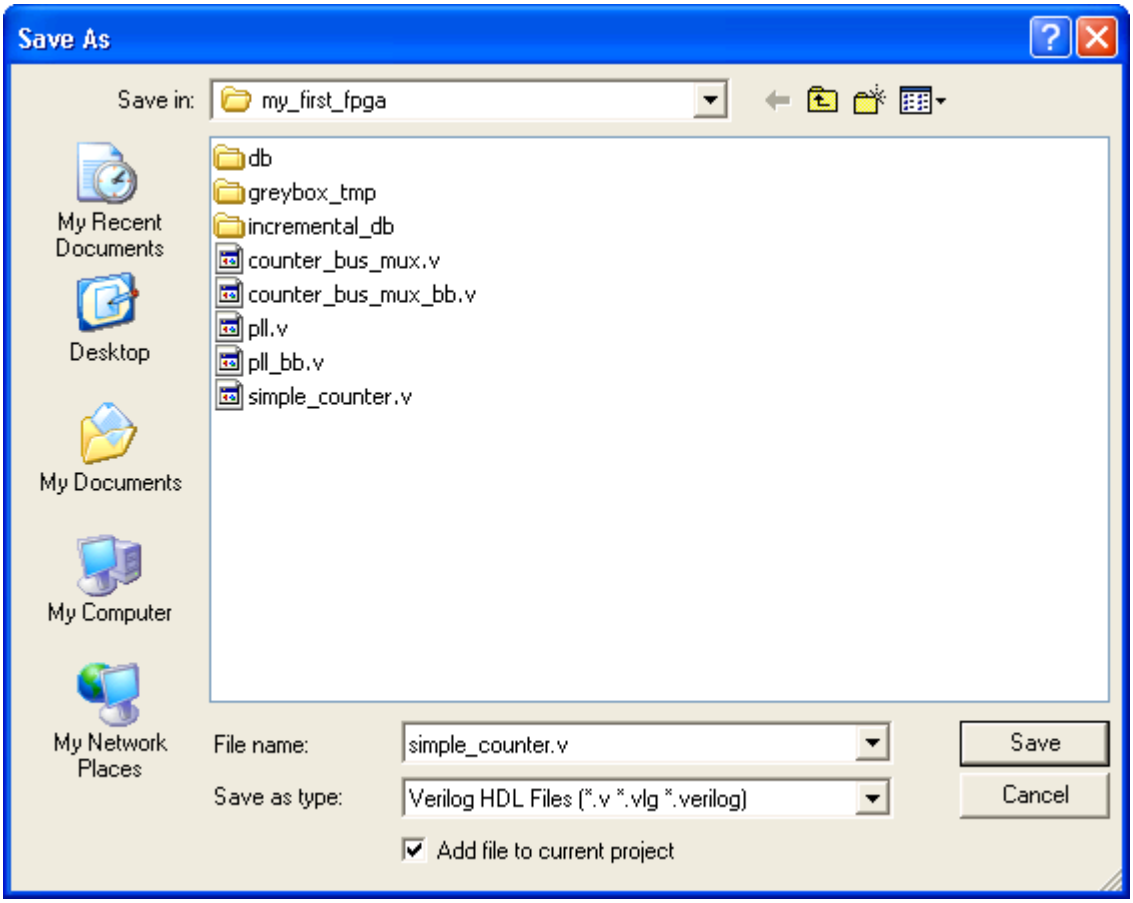

Figure 6-14 Saving the Verilog HDL file

The resulting empty file is ready for you to enter the Verilog HDL code.

4. Type the following Verilog HDL code into the blank simple_counter.v file, as shown in **Figure 6-15**.

```
//It has a single clock input and a 32-bit output port

module simple_counter (
                                    CLOCK_5,
                                    counter_out
                                  );

input     CLOCK_5 ;

output [31:0] counter_out;

reg       [31:0] counter_out;
```

always @ (posedge CLOCK_5) // on positive clock edge

begin

counter_out <= counter_out + 1;// increment counter

end

endmodule // end of module counter

```
1   //It has a single clock input and a 32-bit output port
2   module simple_counter (
3                           CLOCK_5,
4                           counter_out
5                          );
6   input             CLOCK_5 ;
7   output   [31:0]   counter_out;
8   reg      [31:0]   counter_out;
9
10  always @ (posedge CLOCK_5)              // on positive clock edge
11     begin
12        counter_out <= counter_out + 1;   // increment counter
13     end
14  endmodule                               // end of module counter
15
```

Figure 6-15 The Verilog File of simple_counter.v

5. Save the file by choosing **File > Save**, pressing **Ctrl + S**, or by clicking the floppy disk icon.

6. Select **File > Create/Update > Create Symbol Files for Current File** to convert the **simple_counter.v** file to a Symbol File (.sym). You will use this Symbol File to add the HDL code to your schematic.

The Quartus II software creates a Symbol File and displays a message (see **Figure 6-16**).

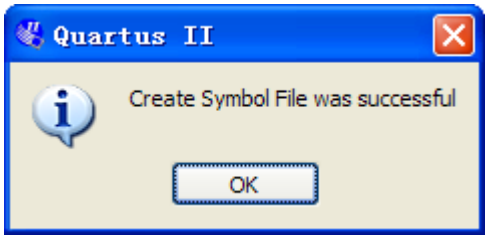

Figure 6-16 Create Symbol File was Successful

7. Click **OK**.

8. To add the **simple_counter.v** symbol to the top-level design, click the **my_first_fpga.bdf** tab.

9. Right click in the blank area of the BDF file, and select **Insert > Symbol**.

10. Double-click the Project directory to expand it.

11. Select the newly created simple_counter symbol by clicking its icon.

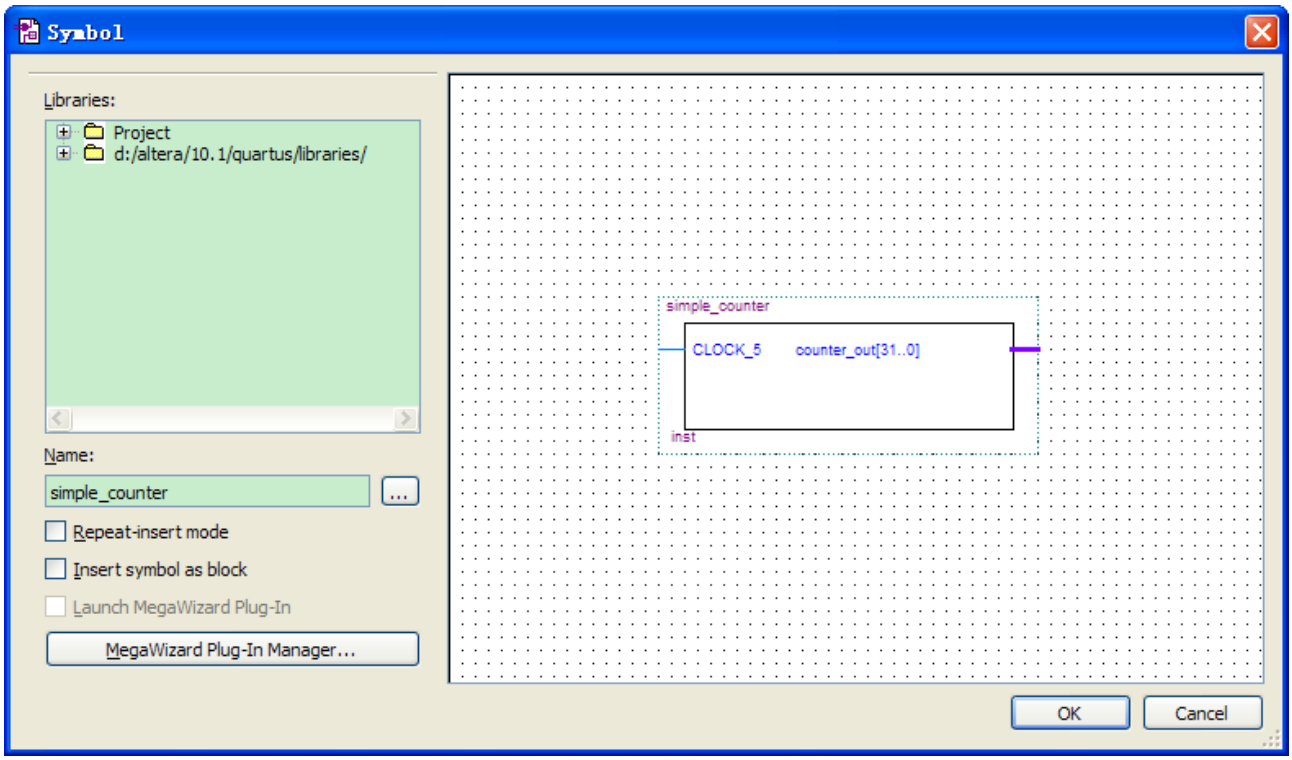

Figure 6-17 Adding the Symbol to the BDF

12. Click **OK**.

13. Move the cursor to the BDF grid; the symbol image moves with the cursor. Click to place the simple_counter symbol onto the BDF. You can move the block after placing it by simply clicking and dragging it to where you want it and releasing the mouse button to place it. See Figure 6-18.

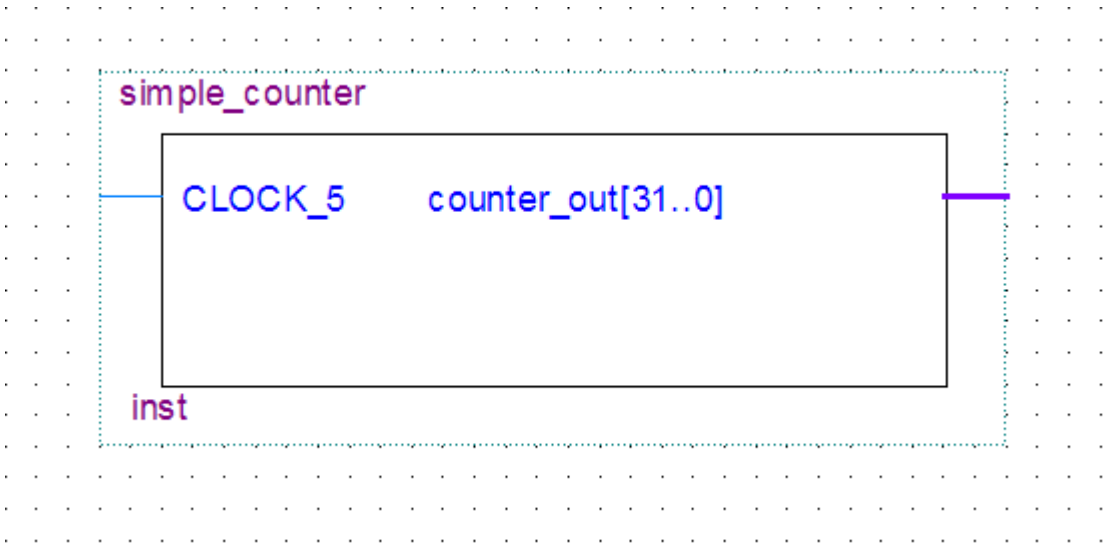

Figure 6-18 Placing the simple_counter symbol

14. Press the **Esc key** or click an empty place on the schematic grid to cancel placing further instances of this symbol.

15. Save your project regularly.

■ Adding a Megafunction to the Schematic

Megafunctions, such as the ones available in the LPM, are pre-designed modules that you can use in FPGA designs. These Altera-provided megafunctions are optimized for speed, area, and device family. You can increase efficiency by using a megafunction instead of writing the function yourself. Altera also provides more complex functions, called MegaCore functions, which you can evaluate for free but require a license file for use in production designs. This tutorial design uses a PLL clock source to drive a simple counter. A PLL uses the on-board oscillator (DE0-Nano Board is 50 MHz) to create a constant clock frequency as the input to the counter. To create the clock source, you will add a pre-built LPM megafunction named ALTPLL.

1. Right click in the blank space in the BDF and select **Insert > Symbol** or click the Add Symbol icon on the toolbar.

2. Click the Megawizard Plug-in Manager button. The MegaWizard® Plug-In Manager appears, as shown in Figure 6-19.

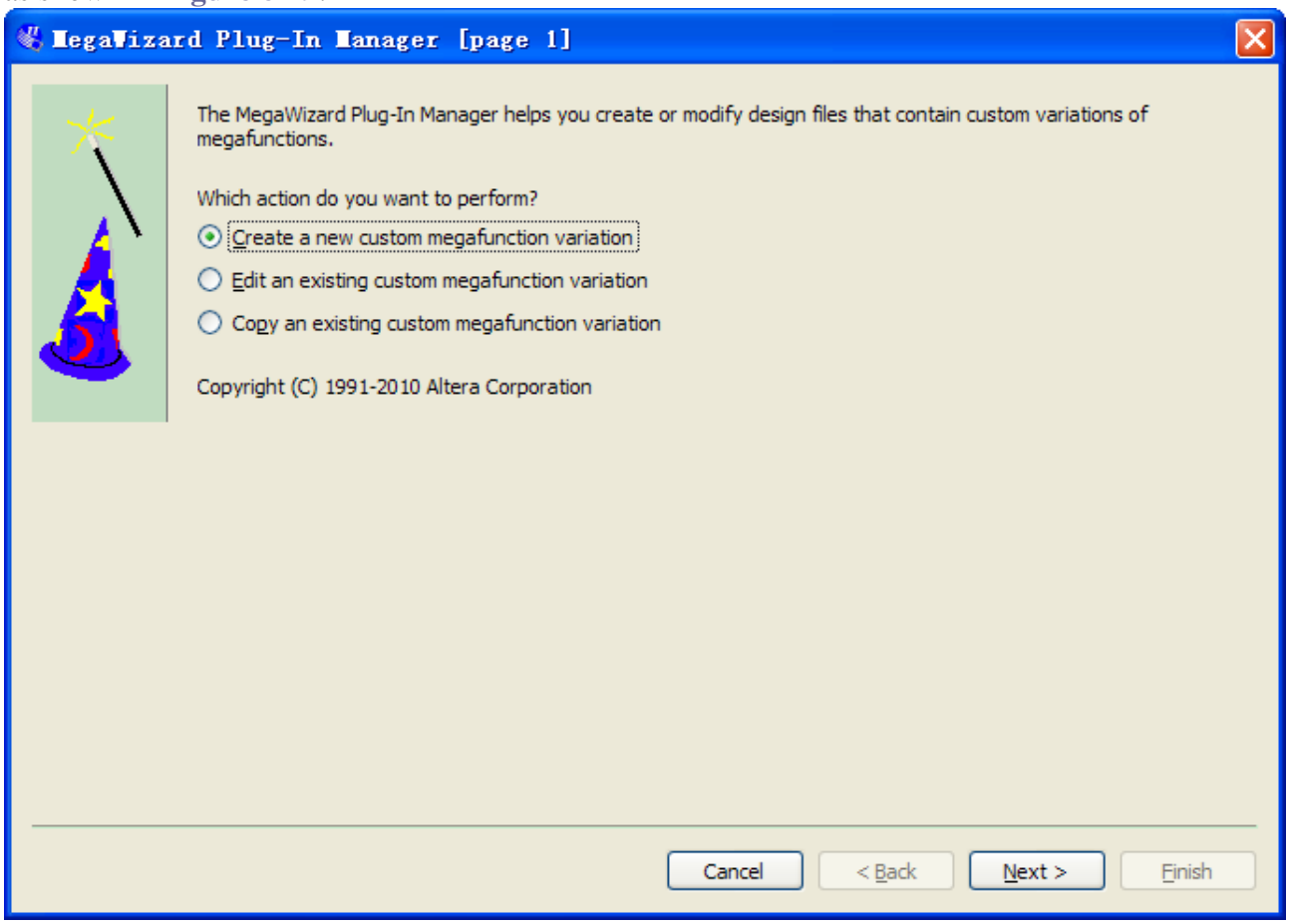

Figure 6-19 Mega Wizard Plug-In Manager

3. Click **Next**.

4. In MegaWizard Plug-In Manager [page 2a], specify the following selections (see **Figure 6-20**):

a. Select **I/O > ALTPLL**.

b. Under "Which device family will you be using?" select the **Cyclone IV E** for DE0-Nano development board.

c. Under "Which type of output file do you want to create?" select **Verilog HDL**.

d. Under "What name do you want for the output file?" type pll at the end of the already created directory name.

e. Click **Next**.

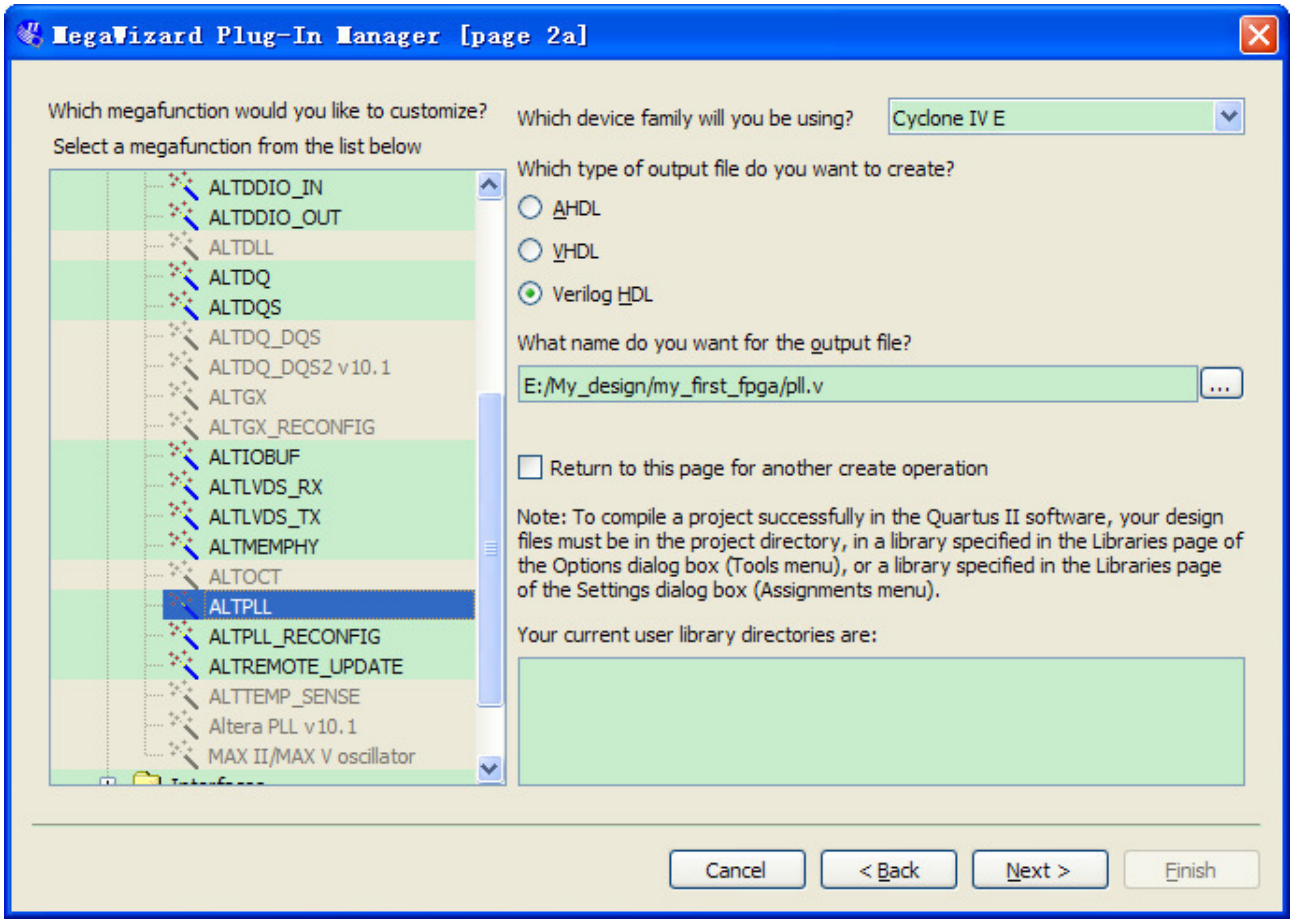

Figure 6-20 MegaWizard Plug-In Manager [page 2a] Selections

5. In the MegaWizard Plug-In Manager [page 3 of 14] window, make the following selections (see **Figure 6-21**).

a. Confirm that the currently selected device family option is set to **Cyclone IV E**.

b. For device speed grade choose 6 for DE0-Nano.

c. Set the frequency of the inclock0 input 50 MHz.

d. Click **Next**.

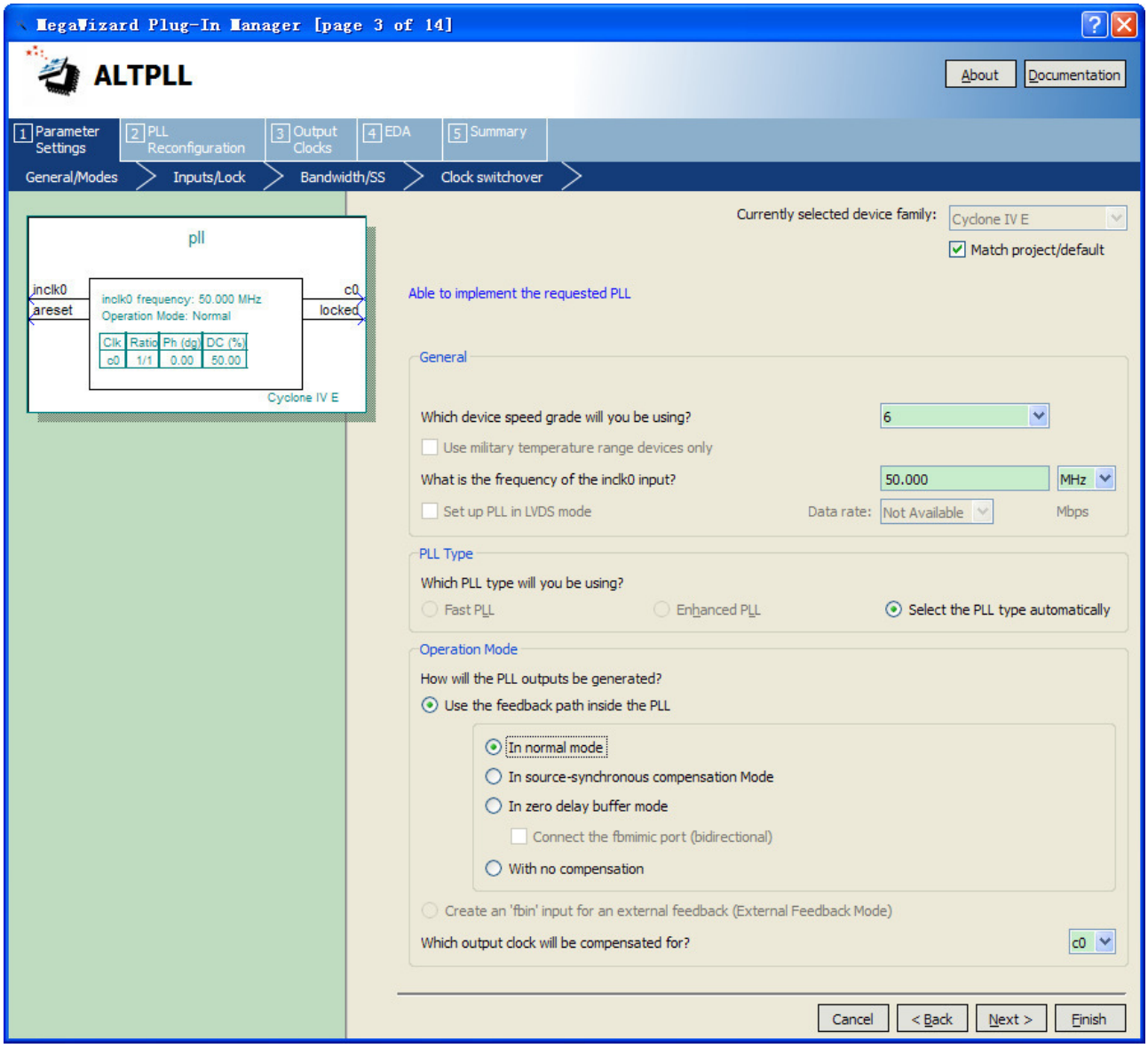

Figure 6-21 MegaWizard Plug-In Manager [page 3 of 14] Selections

6. Unselect all options on MegaWizard page 4. As you turn them off, pins disappear from the PLL block's graphical preview. See **Figure 6-22** for an example.

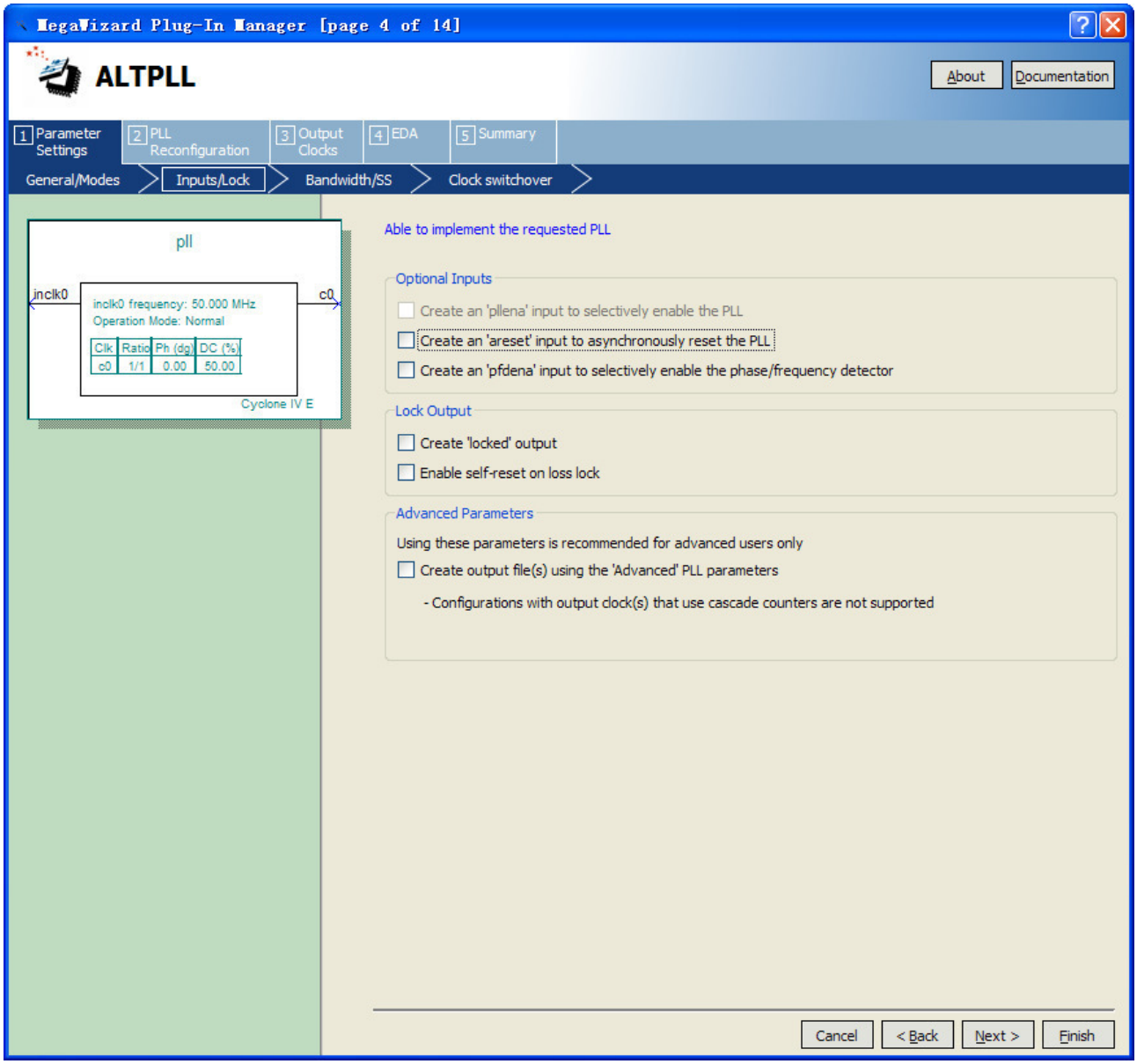

Figure 6-22 MegaWizard Plug-In Manager [page 4 of 14] Selections

7. Click **Next** four times to get to page 8.
8. Set the Clock division factor to 10, as shown in Figure 6-23.

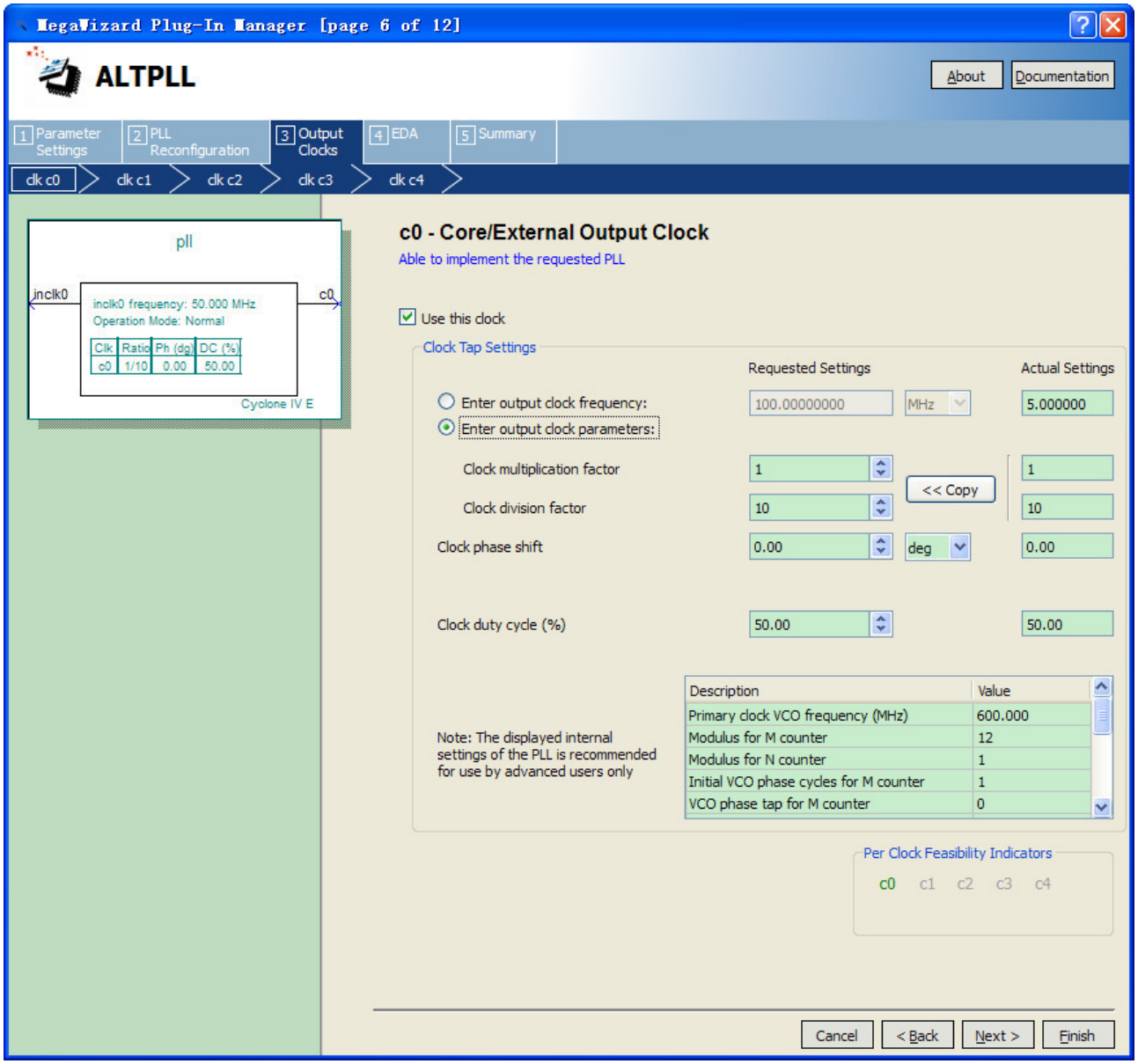

Figure 6-23 MegaWizard Plug-In Manager [page 8 of 14] Selections

9. Click **Next** and then click **Finish**.

10. The wizard displays a summary of the files it creates (see **Figure 6-24**). Select the pll.bsf option and click Finish again.

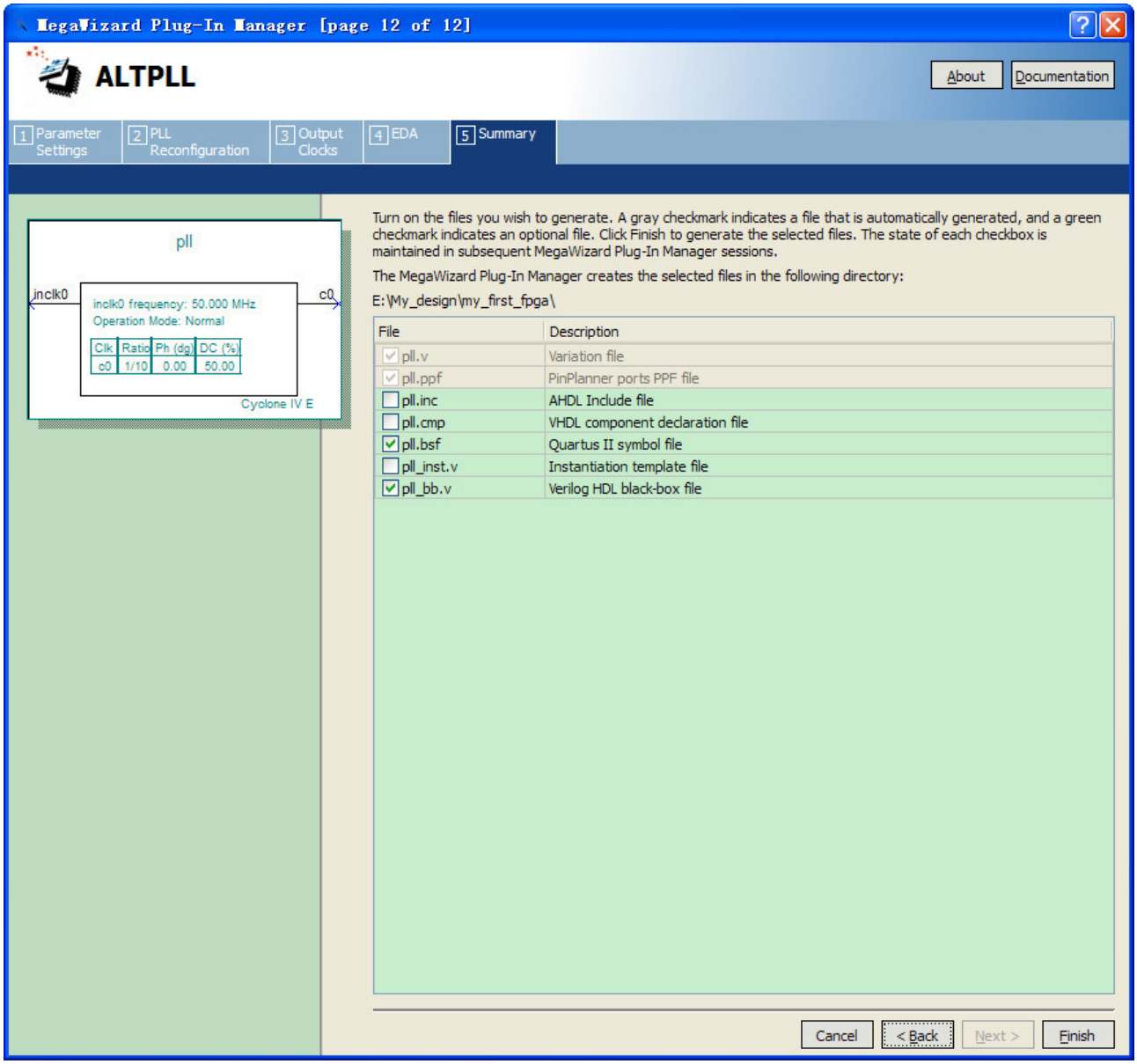

Figure 6-24 Wizard-Created Files

The Symbol window opens, showing the newly created PLL megafunction,a s shown in **Figure 6-25**.

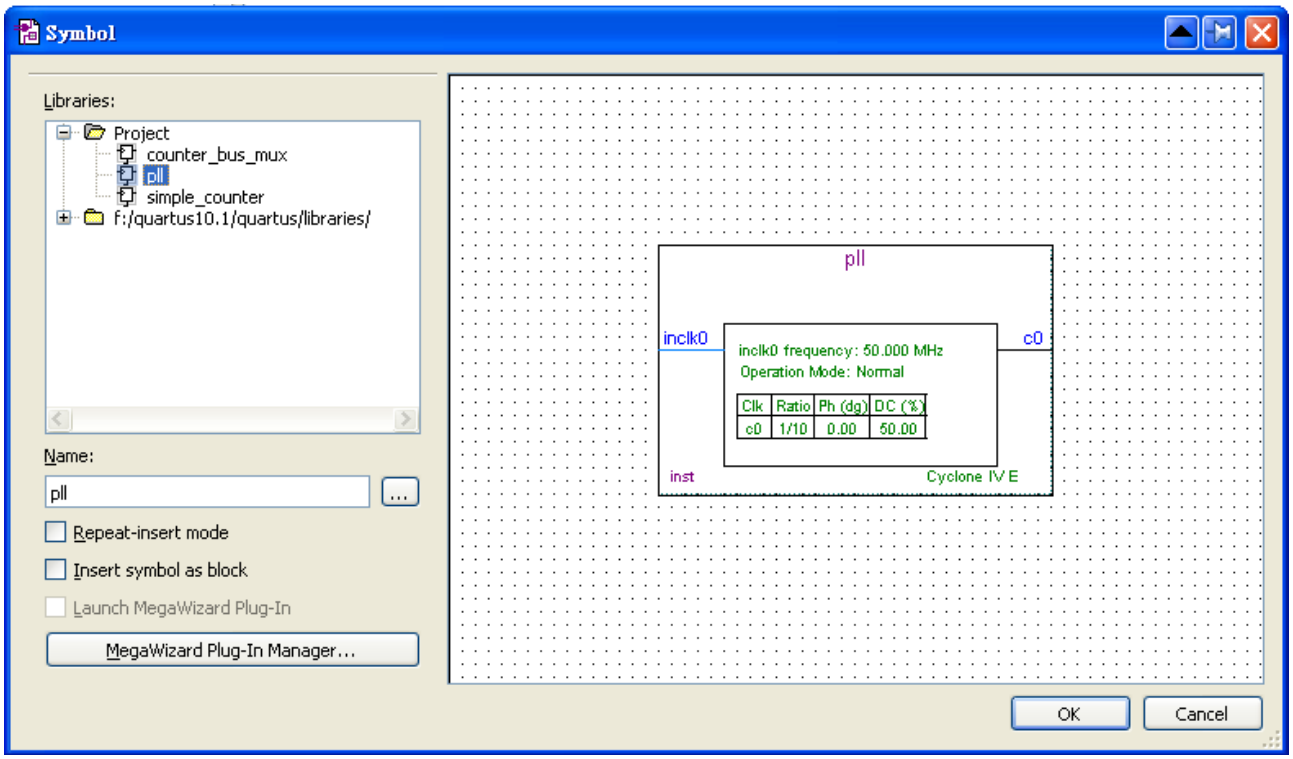

Figure 6-25 PLL Symbol

11. Click **OK** and place the pll symbol onto the BDF to the left of the simple_counter symbol. You can drag and drop the symbols, if you need to rearrange them. See Figure 6-26.

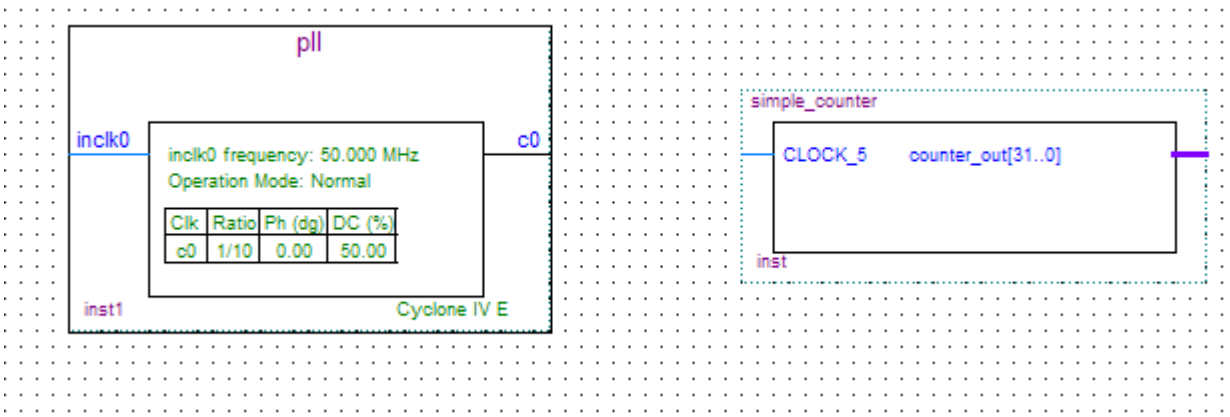

Figure 6-26 Place the PLL Symbol

12. Move the mouse so that the cursor (also called the selection tool) is over the pll symbol's c0 output pin. The orthogonal node tool (cross-hair) icon appears.

13. Click and drag a bus line from the c0 output to the simple_counter clock input. This action ties the pll output to the simple_counter input (see Figure 6-27).

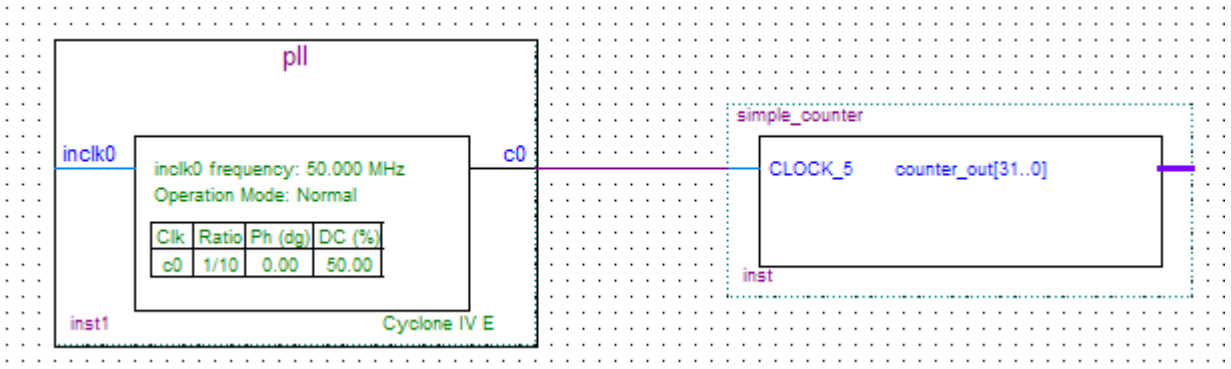

Figure 6-27 Draw a Bus Line connect pll c0 port to simple_counter CLOCK_5 port

■ Adding an Input pin to the Schematic

The following steps describe how to add an input pin to the schematic.

1. Right click in the blank area of the BDF and select **Insert > Symbol**.
2. Under Libraries, select quartus/libraries > primitives > pin >input. See **Figure 6-28**
3. Click OK

If you need more room to place symbols, you can use the vertical and horizontal scroll bars at the edges of the BDF window to view more drawing space.

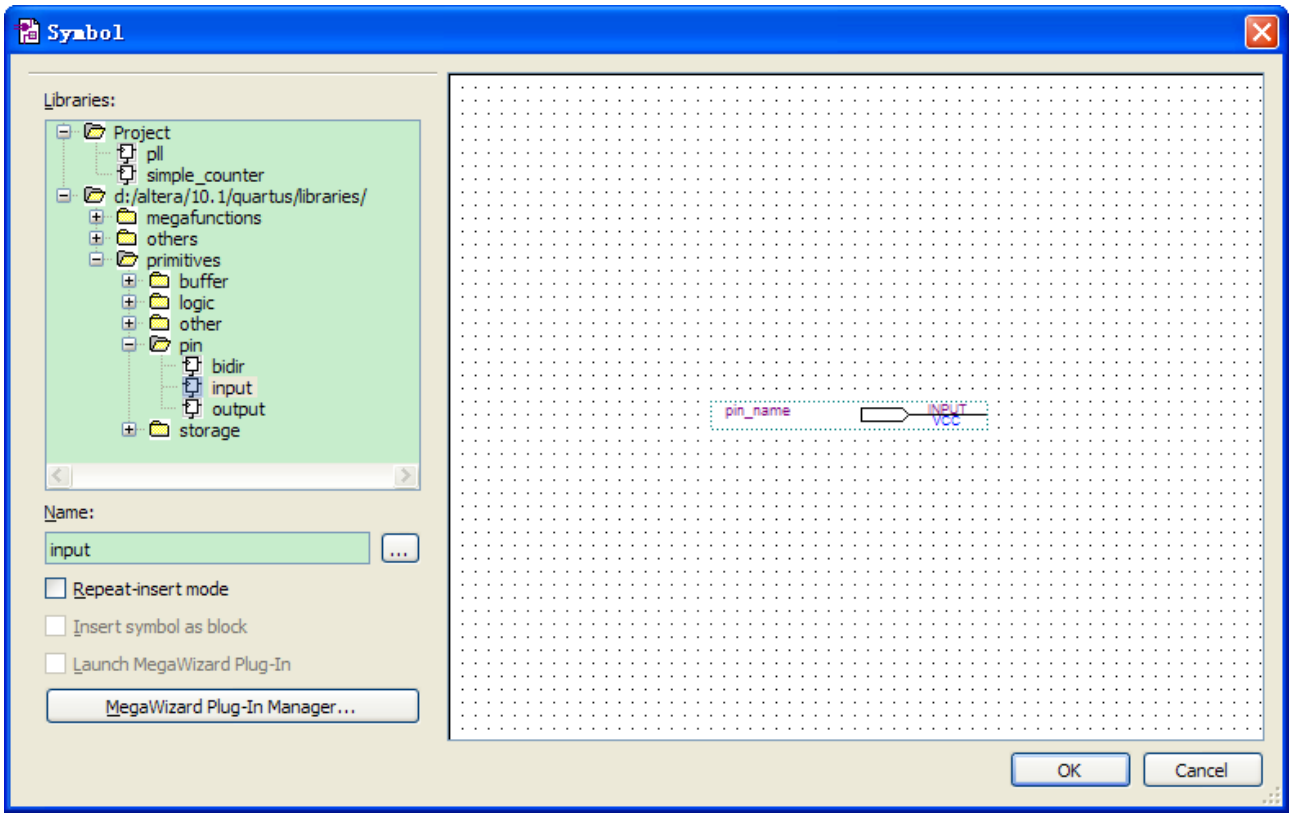

Figure 6-28 Input pin symbol

4. Place the new pin onto the BDF so that it is touching the input to the pll symbol.

5. Use the mouse to click and drag the new input pin to the left; notice that the ports remain connected as shown in Figure 6-29.

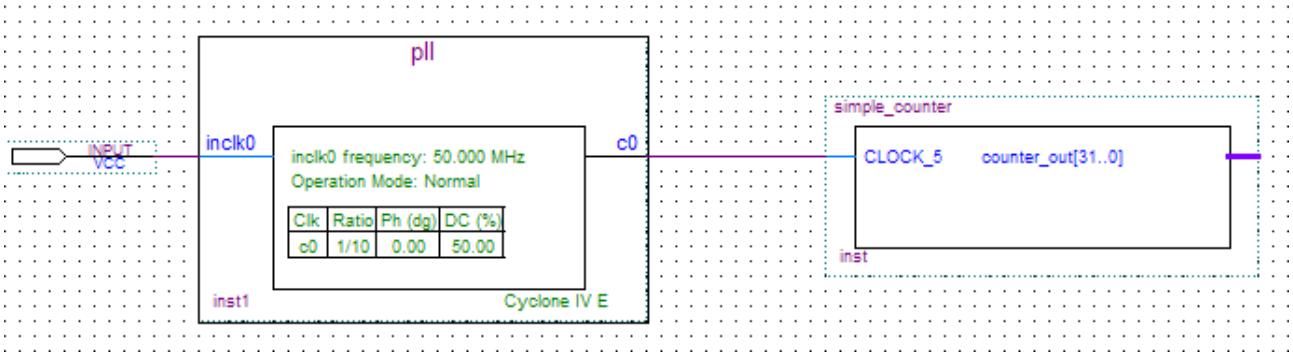

Figure 6-29 Connecting the PLL symbol and Input port

6. Change the pin name by double-clicking pin_name and typing CLOCK_50 (see Figure 6-30). This name correlates to the oscillator clock that is connected to the FPGA.

■ Adding an Output bus to the Schematic

The following steps describe how to add an output bus to the schematic.

1. Using the Orthogonal Bus tool, draw a bus line connected on one side to the simple_counter output port, and leave the other end unconnected at about 4 to 8 grid spaces to the right of the simple_counter.

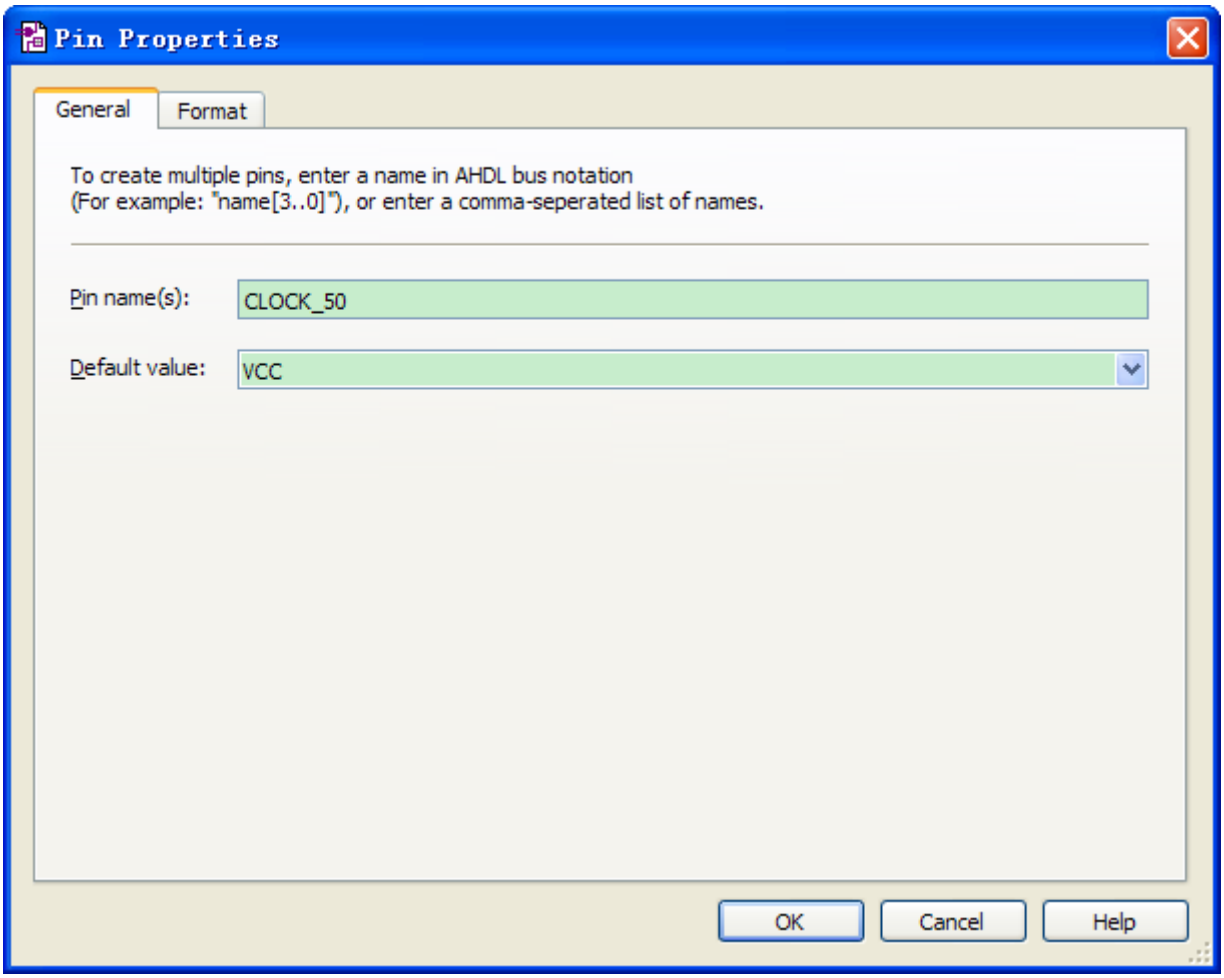

Figure 6-30 Change the input port name

2. Right-click the new output bus line and select **Properties**.

3. Type counter [31..0] as the bus name (see **Figure 6-31**). The notation [X ..Y] is the Quartus II method for specifying the bus width in BDF schematics, where X is the most significant bit (MSB) and Y is the least significant bit (LSB).

4. Click OK. **Figure 6-32** shows the BDF.

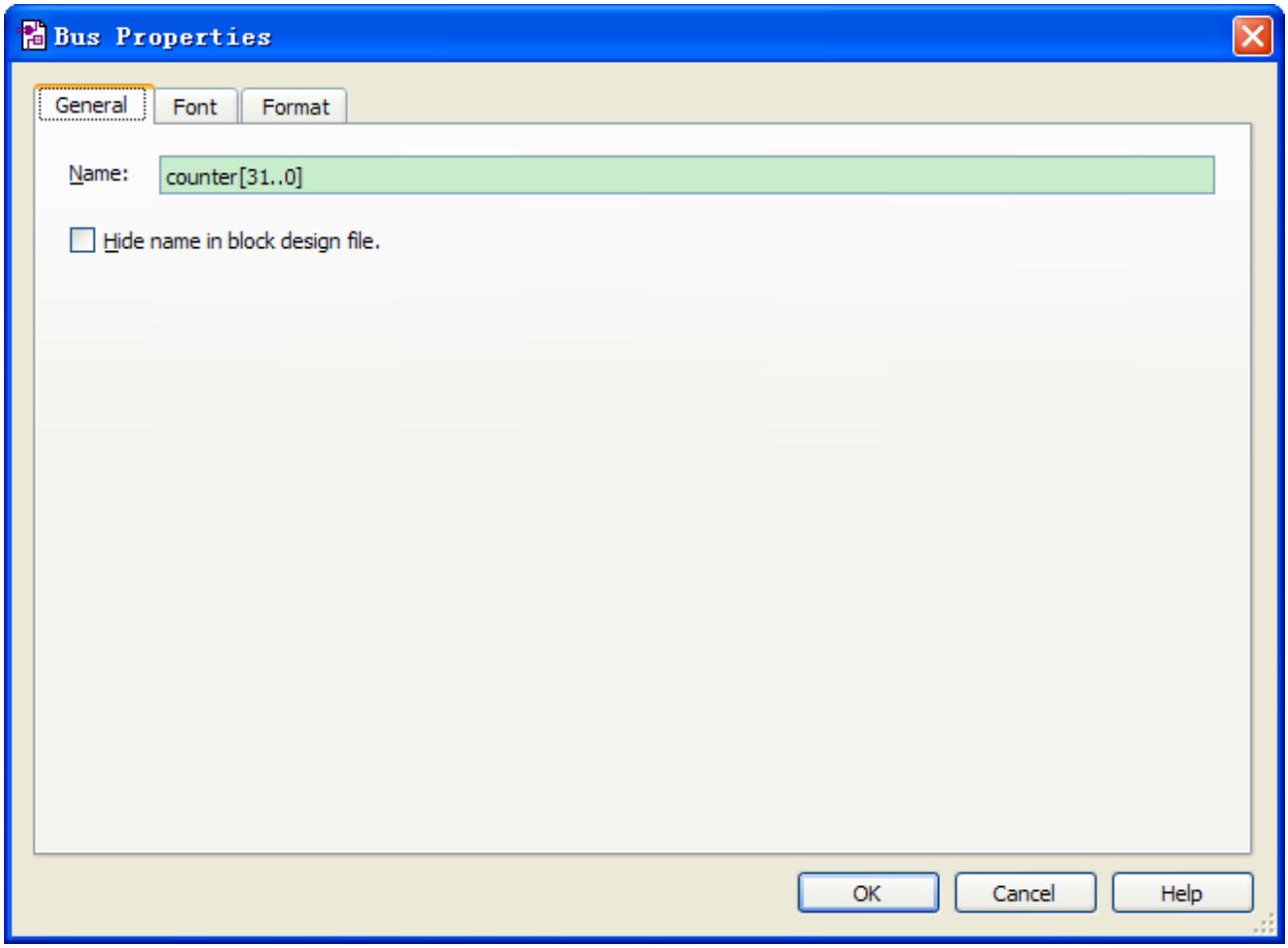

Figure 6-31 Change the output BUS name

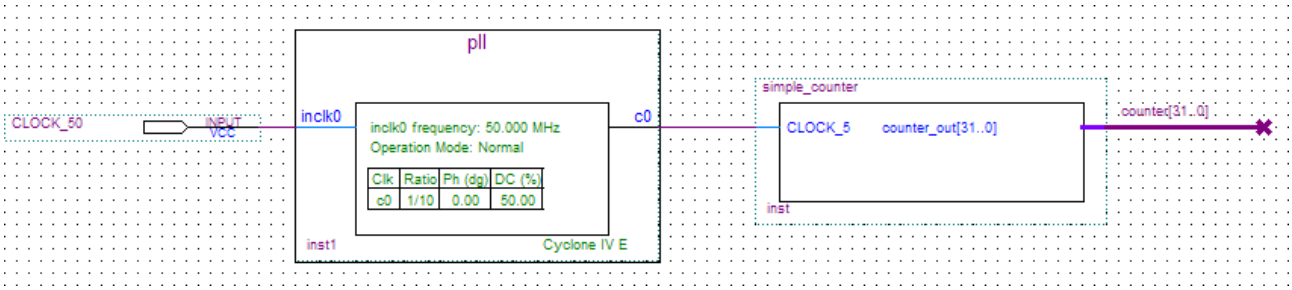

Figure 6-32 Circuit schematic (BDF)

■ Adding a Multiplexer to the Schematic

This design uses a multiplexer to route the simple_counter output to the LED pins on the DE0-Nano development board. You will use the MegaWizard Plug-In Manager to add the multiplexer, lpm_mux. The design multiplexes two portions of the counter bus to four LEDs on the DE0-Nano board. The following steps describe how to add a multiplexer to the schematic.

1. Right click in the blank area of the BDF and select **Insert > Symbol**.
2. Click Megawizard Plug-in Manager.
3. Click **Next**.
4. Select Installed Plug-Ins > Gates > LPM_MUX.
5. Select the **Cyclone IV E** device family, **Verilog HDL** as the output file type, and name the output file **counter_bus_mux.v**, as shown in Figure 6-33.
6. Click **Next**.

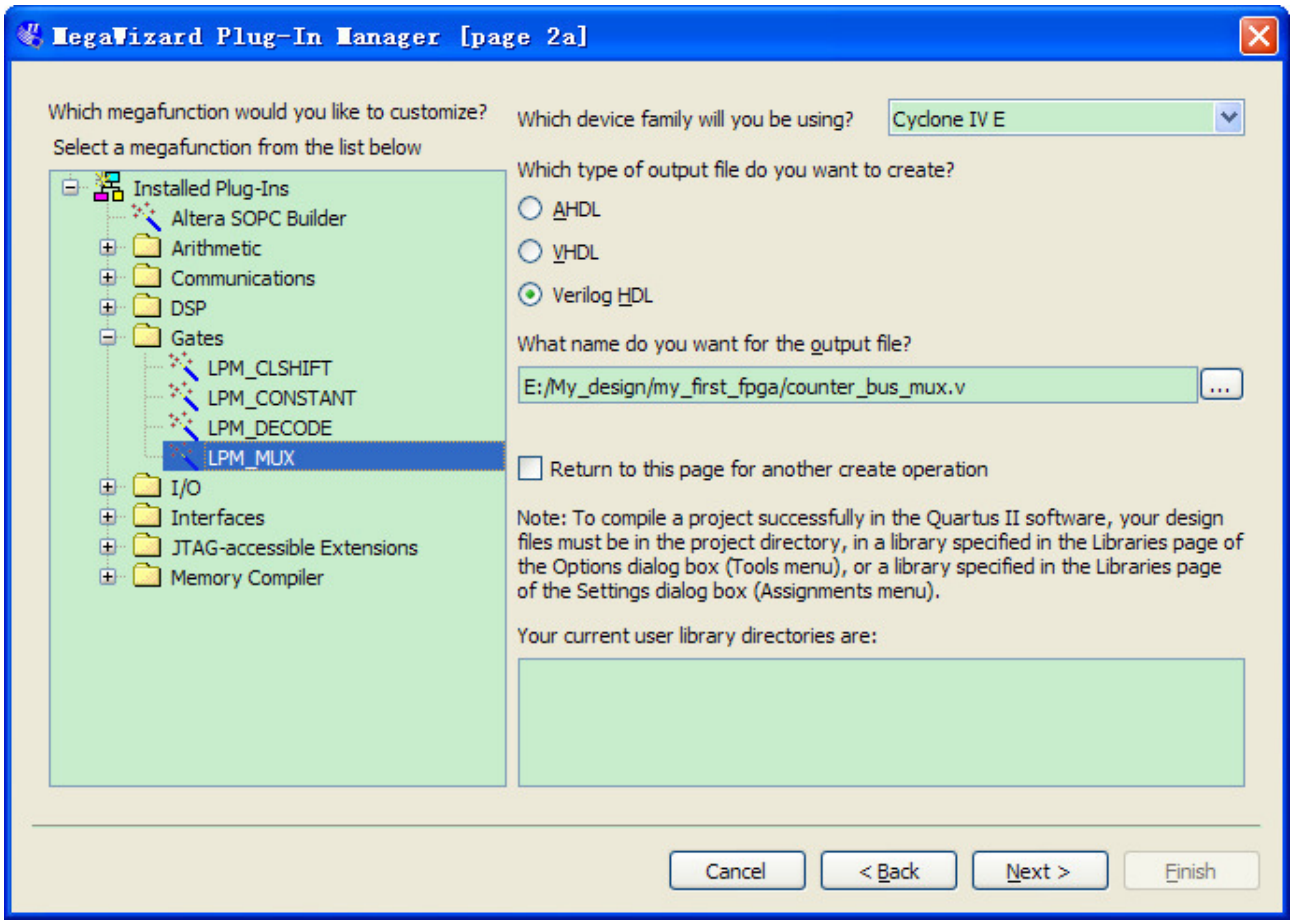

Figure 6-33 Selecting lpm_mux

7. Under "How many 'data' inputs do you want?" select 2 inputs (default).
8. Under "How wide should the 'data' input and the 'result' output buses be?" select 4, as shown in Figure 6-34.

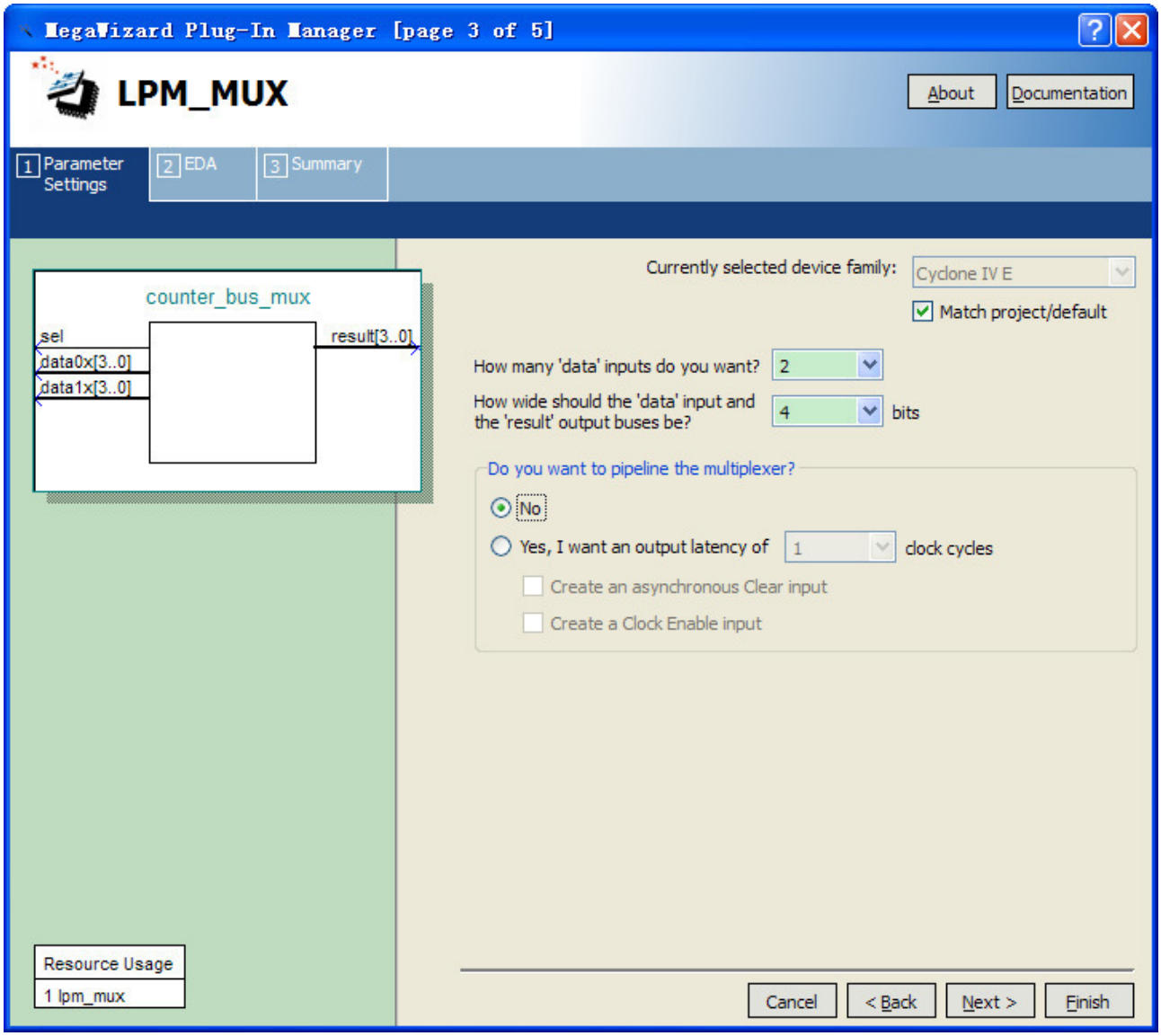

Figure 6-34 lpm_mux settings

9. Click **Next**.
10. Click **Next**.
11. Select the **counter_bus_mux.bsf** option.
12. Click **Finish**. The Symbol window appears (see **Figure 6-35** for an example).

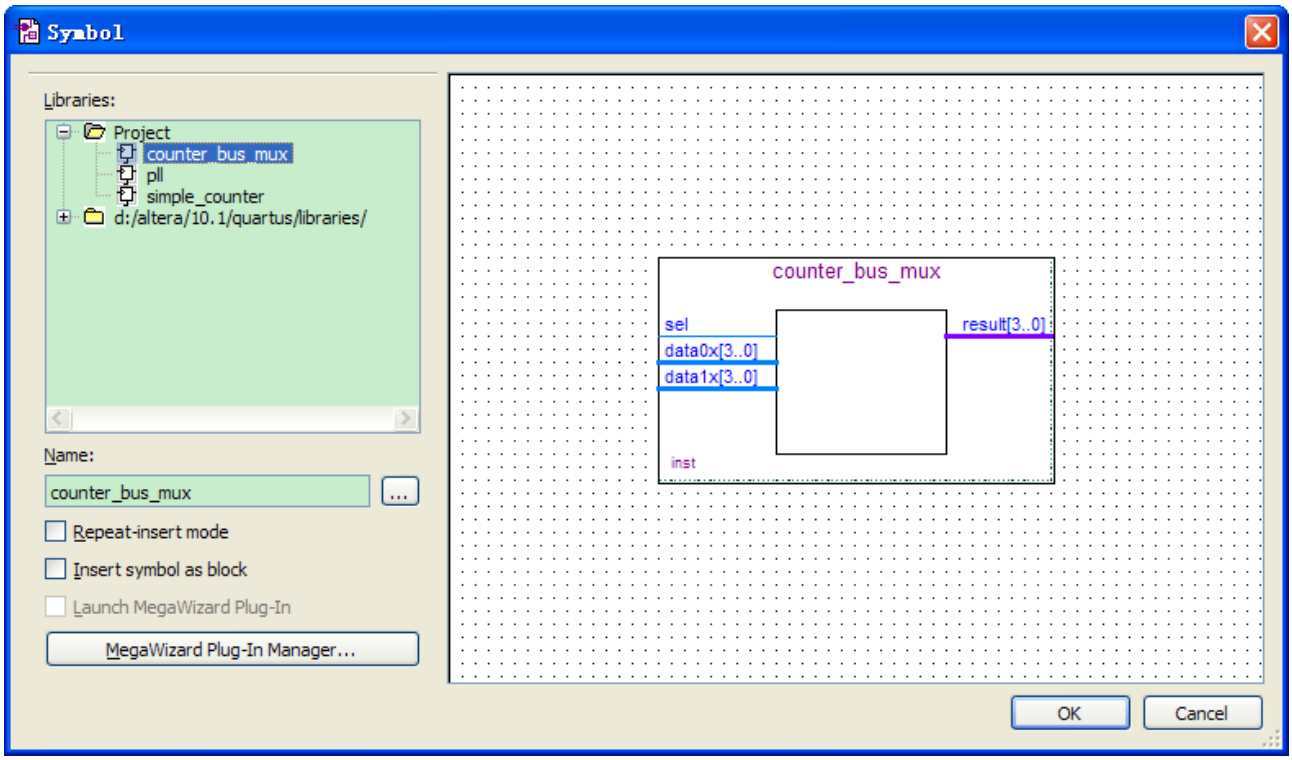

Figure 6-35 lpm_mux Symbol

13. Click **OK**

14. Place the **counter_bus_mux** symbol below the existing symbols on the BDF, as shown in Figure 6-36.

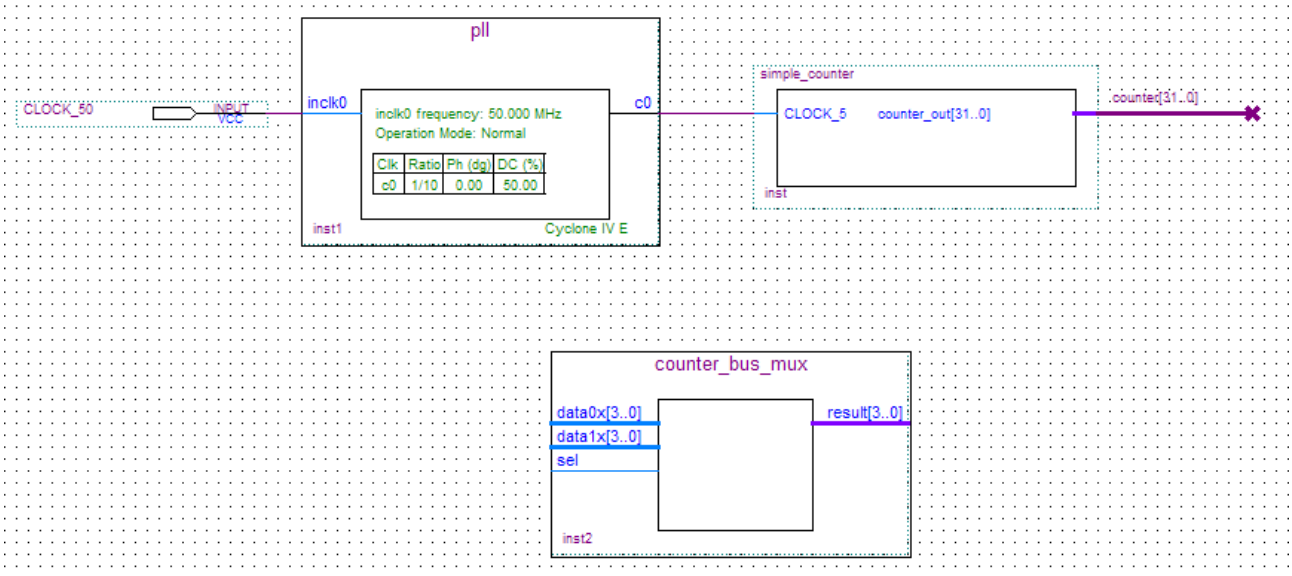

Figure 6-36 Place the lpm_mux symbol

15. Add input buses and output pins to the counter_bus_mux symbol as follows:

a. Using the Orthogonal Bus tool, draw bus lines from the data1x[3..0] and data0x[3..0] input ports to about 8 to 12 grid spaces to the left of counter_bus_mux.

b. Draw a bus line from the result [3..0] output port to about 6 to 8 grid spaces to the right of counter_bus_mux.

c. Right-click the bus line connected to data1x[3..0] and select **Properties**.

d. Name the bus counter[26..23], which selects only those counter output bits to connect to the four bits of the data1x input.

Because the input busses to counter_bus_mux have the same names as the output bus from simple_counter, (counter[x .. y]) the Quartus II software knows to connect these busses.

e. Click **OK**.

f. Right-click the bus line connected to data0x[3..0] and select **Properties**.

g. Name the bus counter [24..21], which selects only those counter output bits to connect to the four bits of the data1x input.

h. Click OK. Figure 6-37 shows the renamed buses.

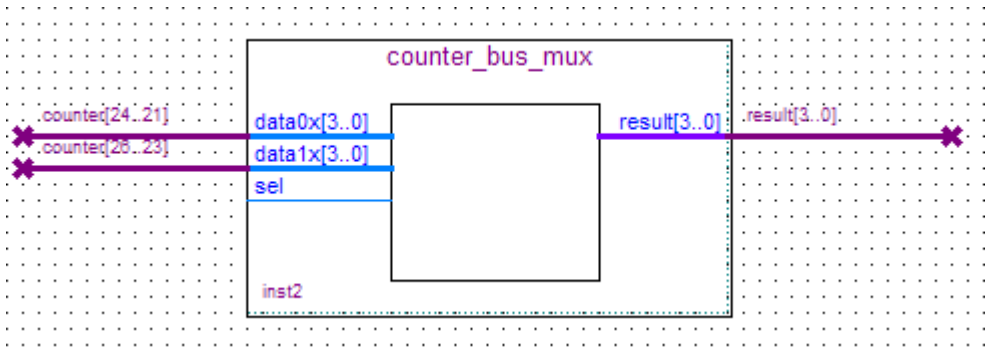

Figure 6-37 Renamed counter_bus_mux Bus Lines

If you have not done so already, you may want to save your project file before continuing.

16. Right click in the blank area of the BDF and select **Insert > Symbol**.

17. Under Libraries, select quartus/libraries > primitives > pin >output, as shown in Figure 6-38.

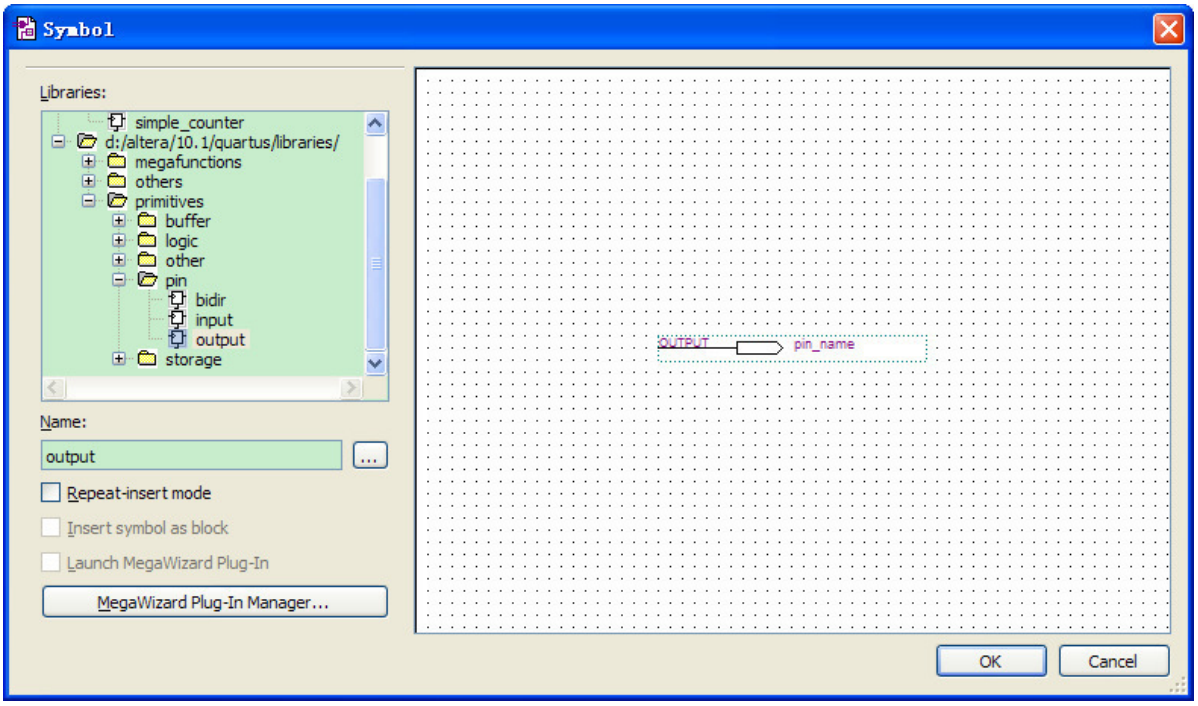

Figure 6-38 Choose output pin

18. Click **OK**.

19. Place this output pin so that it connects to the counter_bus_mux's result [3..0] bus output line.

20. Rename the output pin as LED [3..0]. (see **Figure 6-39**).

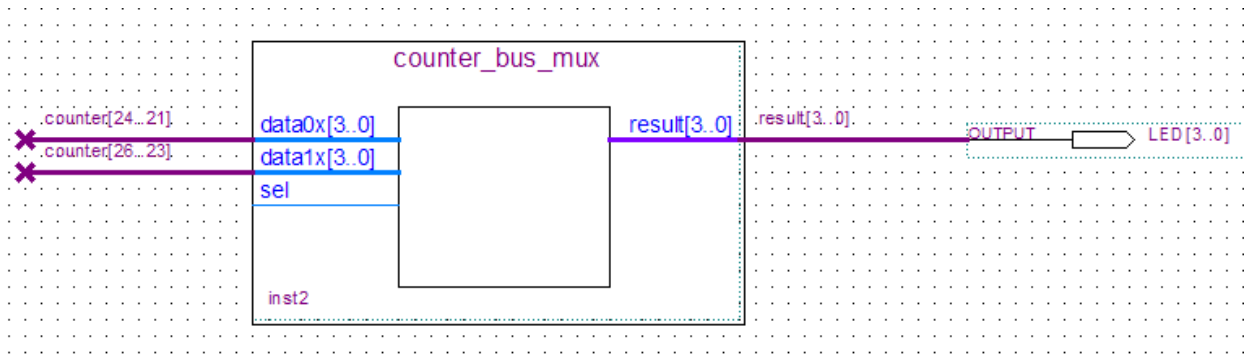

Figure 6-39 Rename the output pin

21. Attach an input pin to the multiplexer select line using an input pin:

a. Right click in the blank area of the BDF and select **Insert > Symbol**.

b. Under Libraries, double-click quartus/libraries/ > primitives > pin > input.

c. Click **OK**.

22. Place this input pin below **counter_bus_mux**.

23. Connect the input pin to the **counter_bus_mux** sel pin.

24. Rename the input pin as KEY [0] (see **Figure 6-40**).

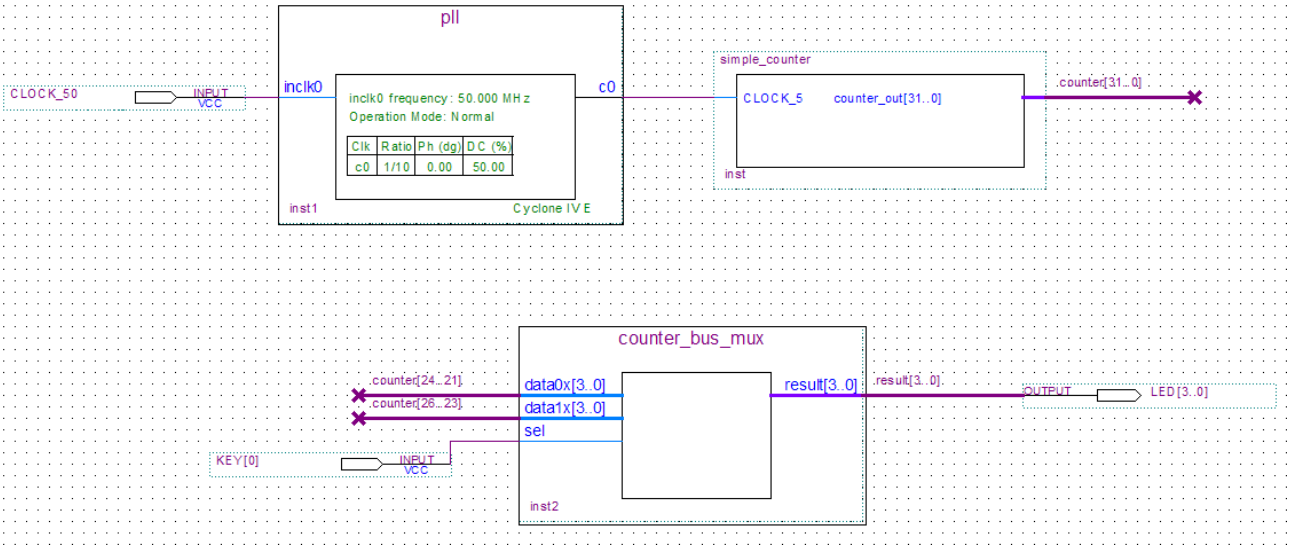

Figure 6-40 Adding the KEY [0] Input Pin

You have finished adding all required components of the circuit to your design. You can add notes or information to the project as text using the Text tool on the toolbar (indicated with the A symbol). For example, you can add the label "OFF = SLOW, ON = FAST" to the KEY [0] input pin and add a project description, such as "DE0-Nano Tutorial Project."

6.6 Assign the Pins

In this section, you will make pin assignments. Before making pin assignments, perform the following steps:

1. Select **Processing > Start > Start Analysis & Elaboration** in preparation for assigning pin locations.

2. Click **OK** in the message window that appears after analysis and elaboration completes.

To make pin assignments to the KEY [0] and CLOCK_50 input pins and to the LED[3..0] output pins, perform the following steps:

1. Select **Assignments > Pin Planner**, which opens the Pin Planner, a spreadsheet-like table of specific pin assignments. The Pin Planner shows the design's six pins. See **Figure 6-41**

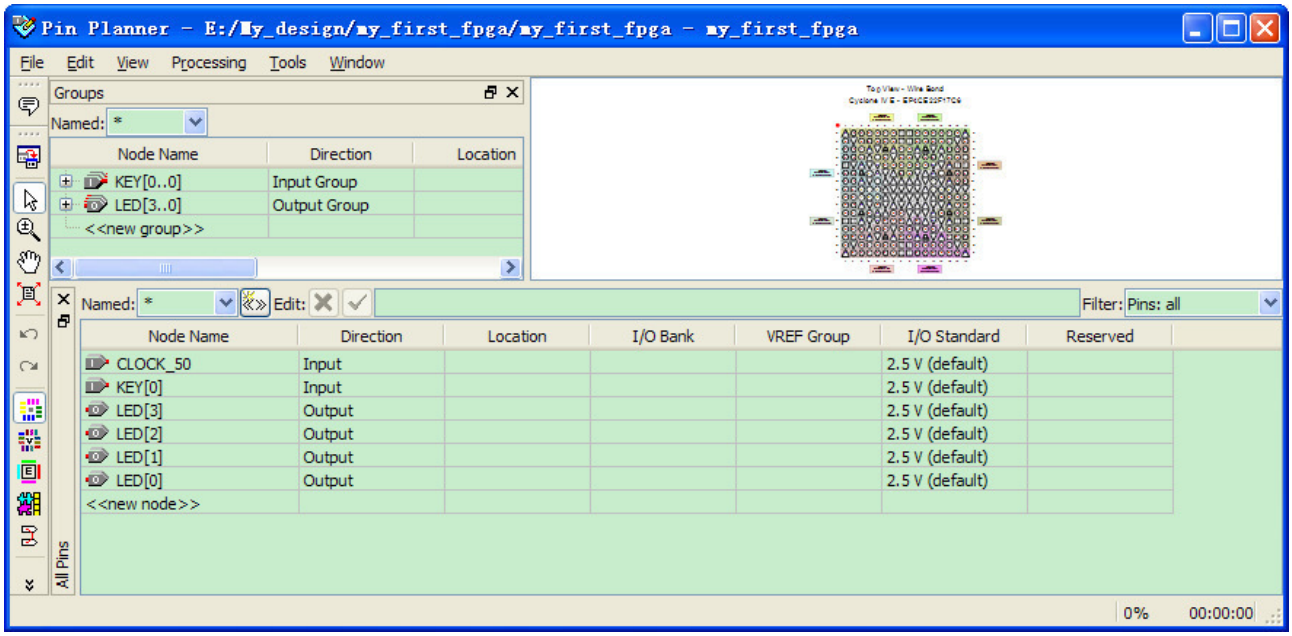

Figure 6-41 Pin Planner Example

2. In the Location column next to each of the six node names, add the coordinates (pin numbers) as shown in Table 6-1 for the actual values to use with your DE0-Nano board.

Table 6-1 Pin Information Setting

Pin Name	*FPGA Pin Location*
KEY[0]	**J15**
LED[3]	**A11**
LED[2]	**B13**
LED [1]	**A13**
LED [0]	**A15**
CLOCK_50	**R8**

Double-click in the Location column for any of the six pins to open a drop-down list and type the location shown in the table. Alternatively, you can select the pin from a drop-down list. For example, if you type **F1** and press the **Enter** key, the Quartus II software fills in the full PIN_F1 location name for you. The software also keeps track of corresponding FPGA data such as the I/O bank and VREF Group. Each bank has a distinct color, which corresponds to the top-view wire bond drawing in the upper right window, as shown in Figure 6-42.

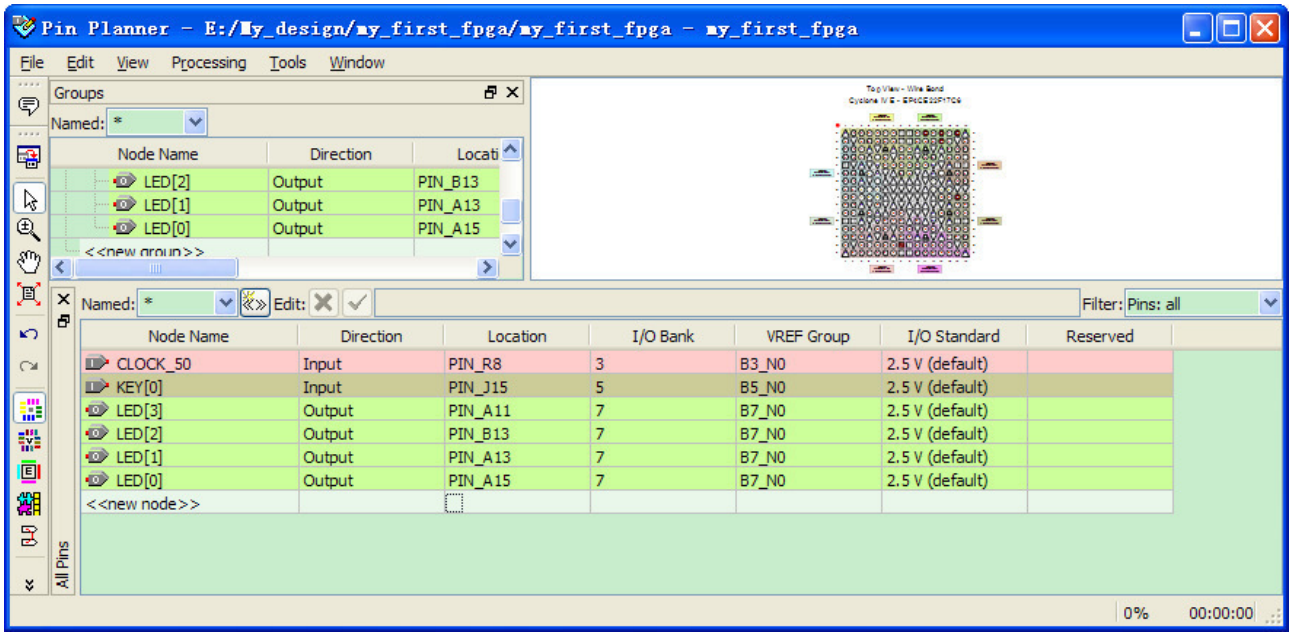

Figure 6-42 Completed Pin Planning Example

Now, you are finished creating your Quartus II design!

6.7 Create a Default TimeQuest SDC File

Timing settings are critically important for a successful design. For this tutorial you will create a basic Synopsys Design Constraints File (.sdc) that the Quartus II TimeQuest Timing Analyzer uses during design compilation. For more complex designs, you will need to consider the timing requirements more carefully.

To create an SDC, perform the following steps:

1. Open the TimeQuest Timing Analyzer by choosing **Tools > TimeQuest Timing Analyzer**.
2. Select **File > New SDC file**. The SDC editor opens.
3. Type the following code into the editor:

```
create_clock -period 20.000 -name CLOCK_50

derive_pll_clocks

derive_clock_uncertainty
```

4. Save this file as my_first_fpga.sdc (see Figure 6-43)

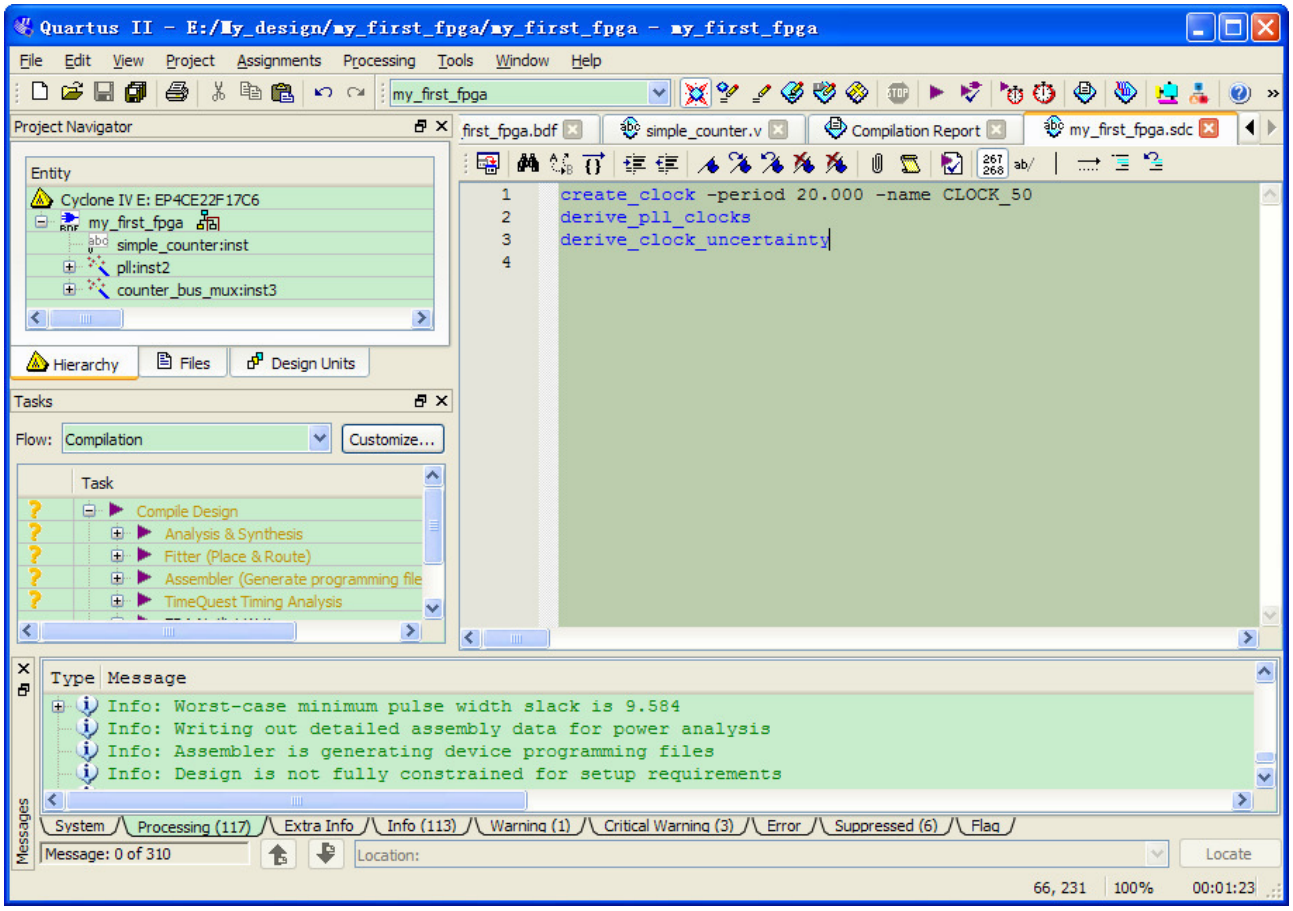

Figure 6-43 Default SDC

Naming the SDC with the same name as the top-level file causes the Quartus II software to use this timing analysis file automatically by default. If you used another name, you would need to add the SDC to the Quartus II assignments file.

6.8 Compile Your Design

After creating your design you must compile it. Compilation converts the design into a bitstream that can be downloaded into the FPGA. The most important output of compilation is an SRAM Object File (.sof), which you use to program the device. Also, the software generates report files that provide information about your circuit as it compiles.

Now that you have created a complete Quartus II project and entered all assignments, you can compile the design.

In the **Processing** menu, select **Start Compilation** or click the **Play** button on the toolbar.

If you are asked to save changes to your BDF, click **Yes**.

While compiling your design, the Quartus II software provides useful information about the compilation, as shown in Figure 6-44.

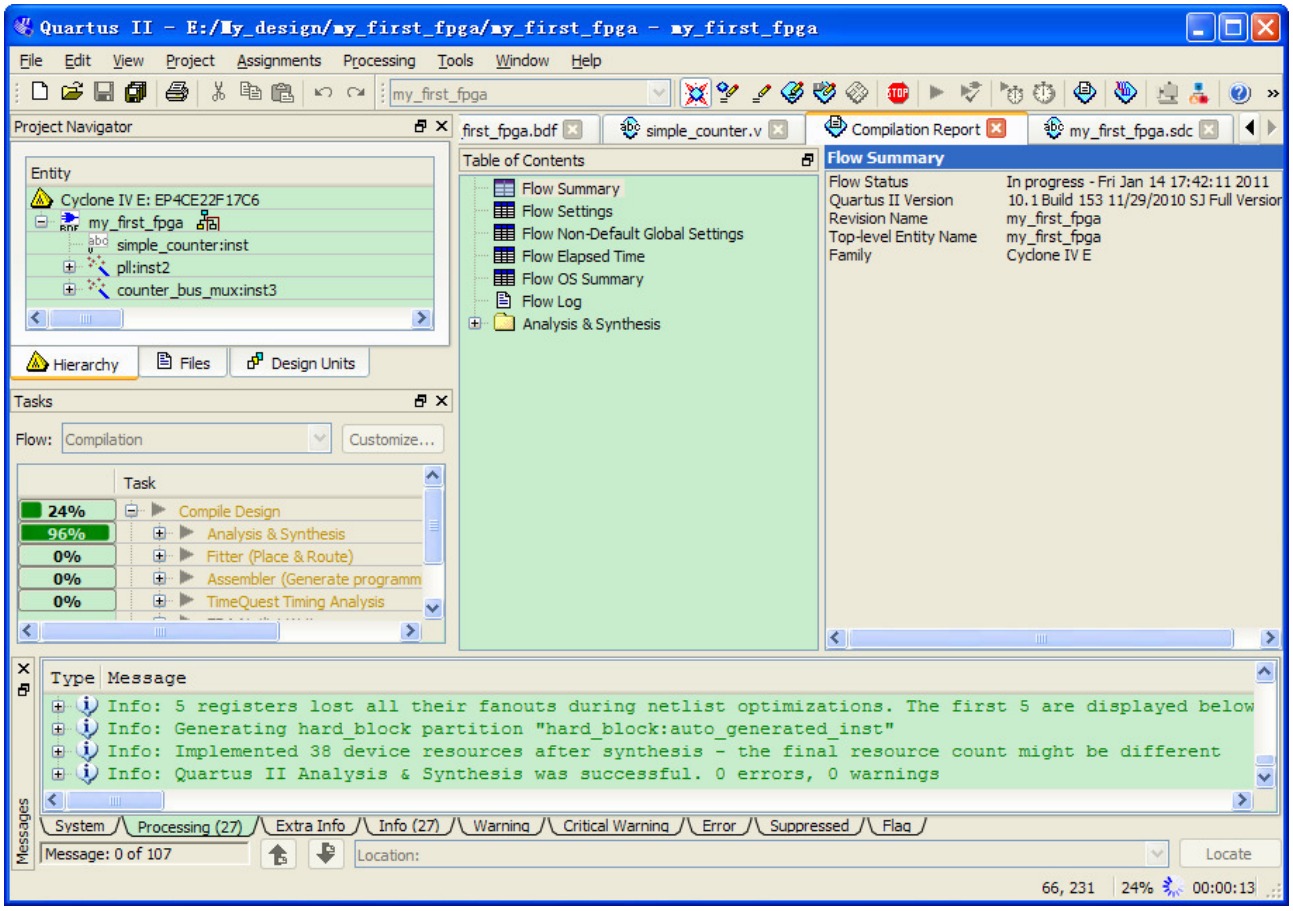

Figure 6-44 Compilation Message for project

When compilation is complete, the Quartus II software displays a message. Click OK to close the message box.

The Quartus II Messages window displays many messages during compilation. It should not display any critical warnings; it may display a few warnings that indicate that the device timing information is preliminary or that some parameters on the I/O pins used for the LEDs were not set. The software provides the compilation results in the Compilation Report tab as shown in **Figure 6-45**.

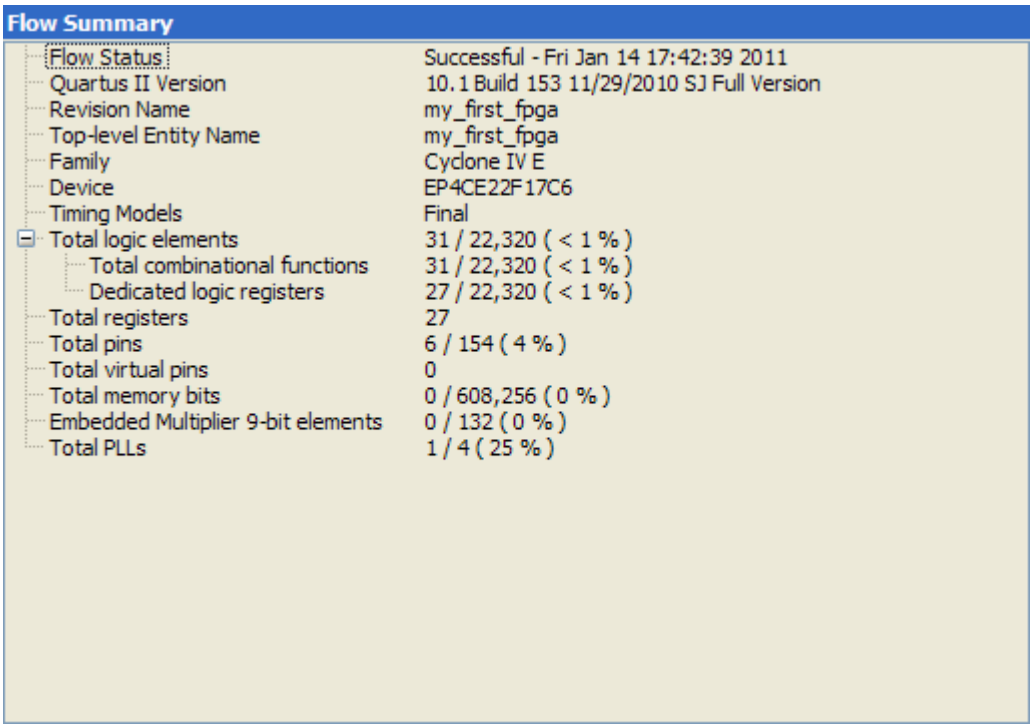

Figure 6-45 Compilation Report Example

6.9 Program the FPGA Device

After compiling and verifying your design you are ready to program the FPGA on the development board. You download the SOF you just created into the FPGA using the USB-Blaster circuitry on the board. Set up your hardware for programming using the following steps:

First, connect the USB cable, which was included in your development kit, between the DE0-Nano and the host computer. Refer to the getting started user guide for detailed instructions on how to connect the cables.

Refer to the getting started user guide for detailed instructions on how to connect the cables.

Program the FPGA using the following steps.

1. Select Tools > Programmer. The Programmer window opens, as shown in **Figure 6-46**.

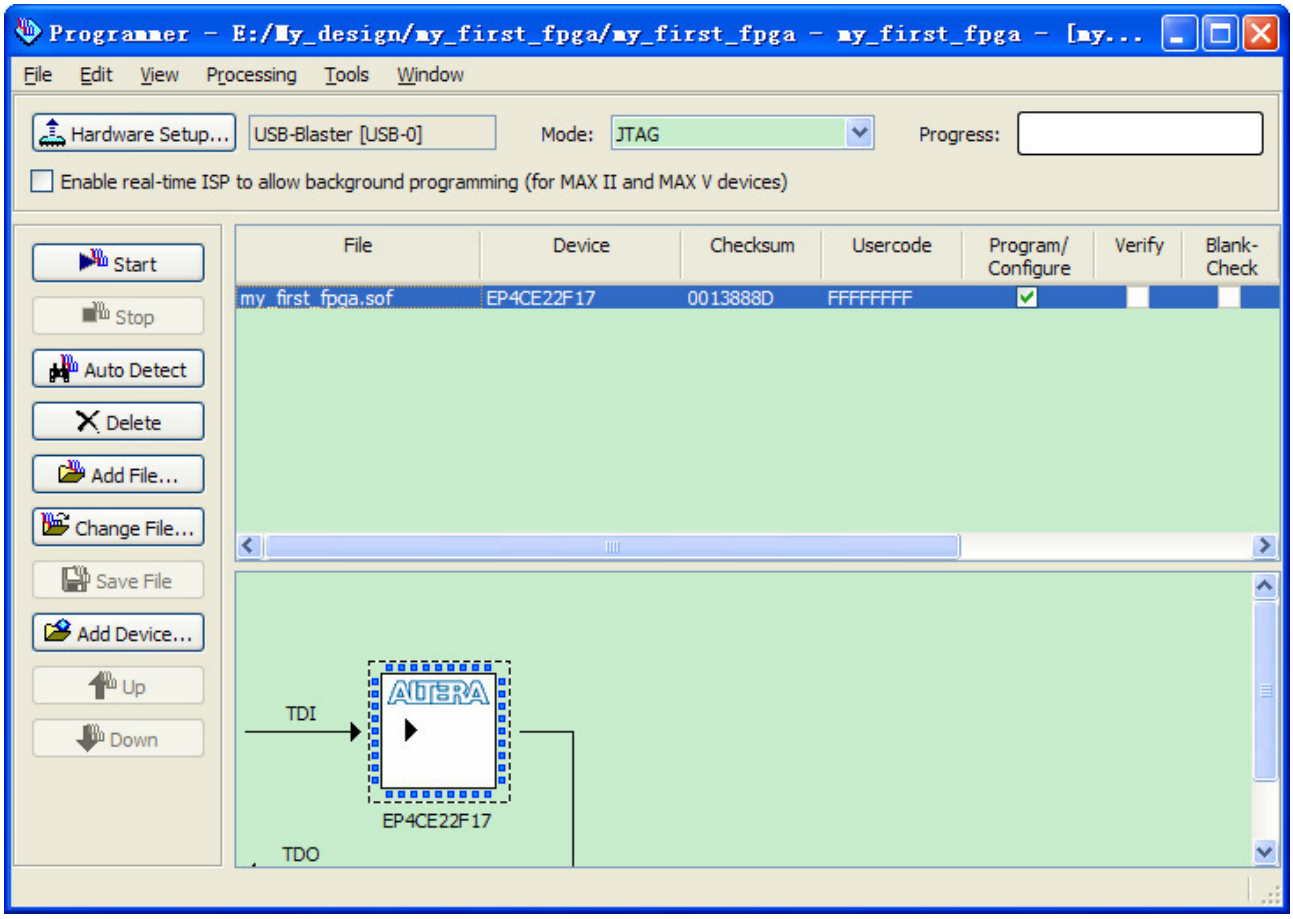

Figure 6-46 Programmer Window

2. Click Hardware Setup.

3. If it is not already turned on, turn on the USB-Blaster [USB-0] option under currently selected hardware, as shown in Figure 6-47.

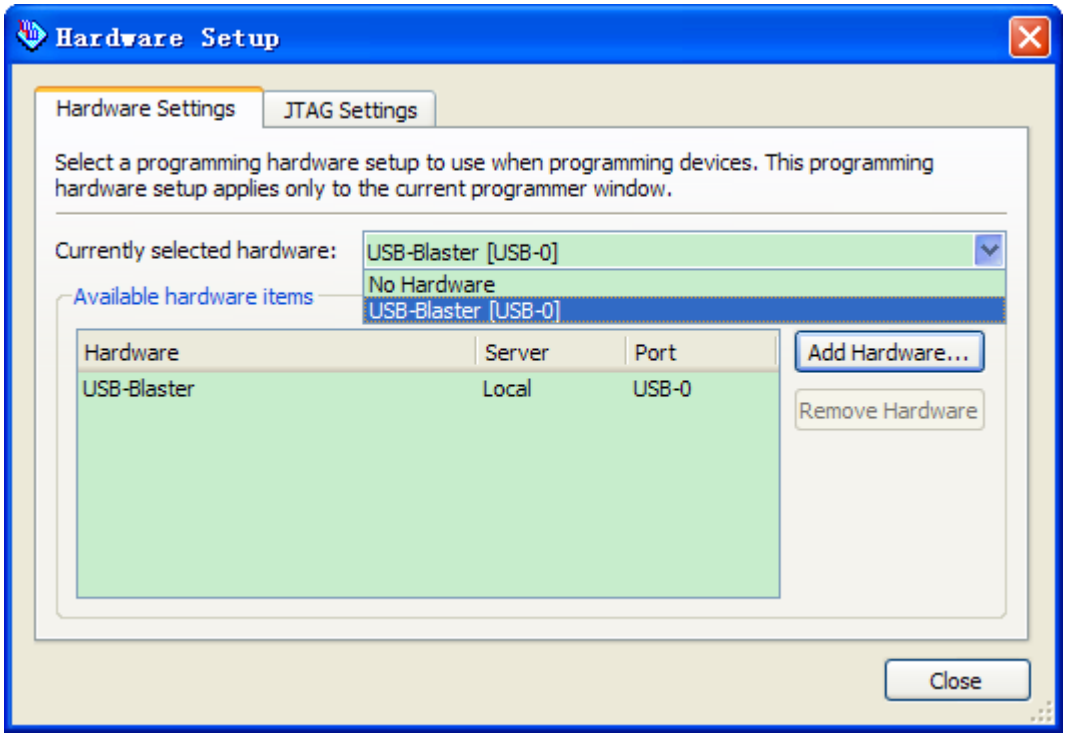

Figure 6-47 Hardware Setting

4. Click **Close**.
5. If the file name in the Programmer does not show **my_first_fpga.sof**, click **Add File**.
6. Select the my_first_fpga.sof file from the project directory (see Figure 6-48).
7. Click the **Start** button.

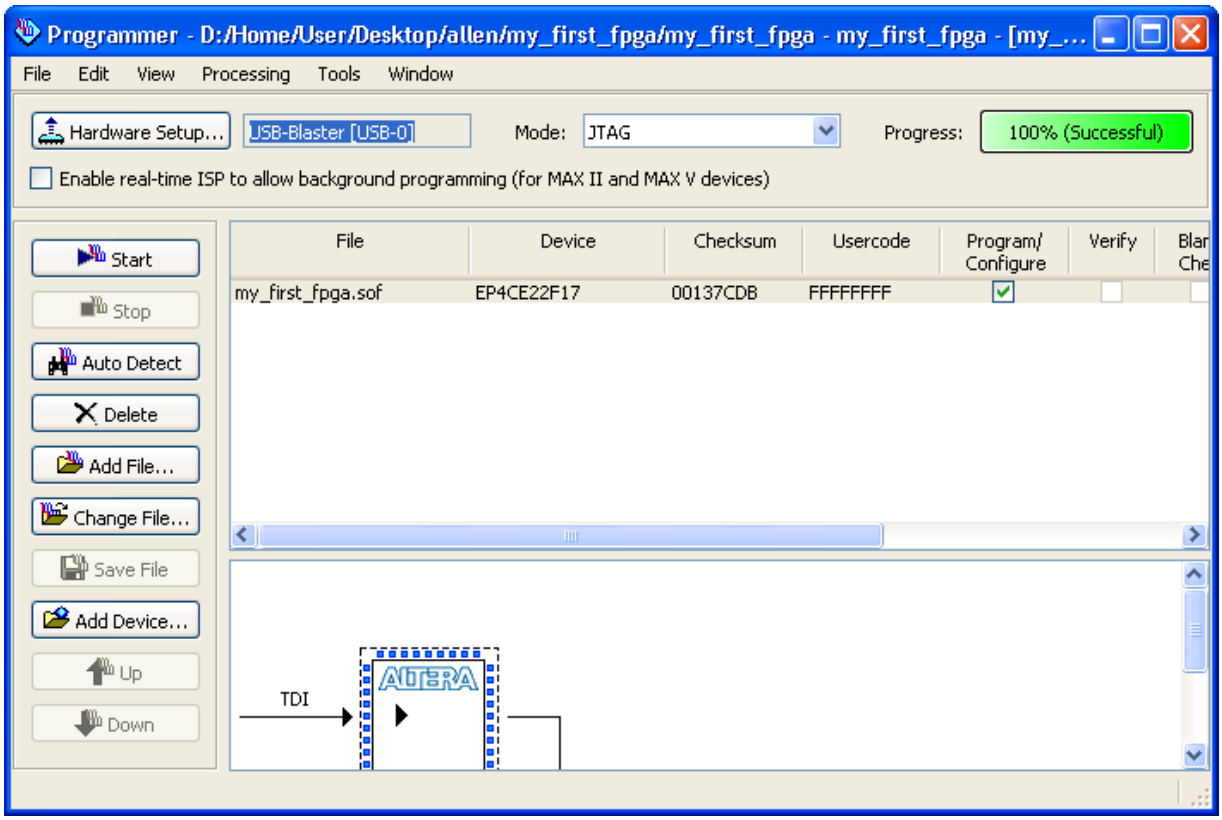

Figure 6-48 Downloading Complete

Congratulations, you have created, compiled, and programmed your first FPGA design! The compiled SRAM Object File (.sof) is loaded onto the FPGA on the development board and the design should be running.

6.10 Verify The Hardware

When you verify the design in hardware, you observe the runtime behavior of the FPGA hardware design and ensure that it is functioning appropriately.

Verify the design by performing the following steps:

1. Observe that the four development board LEDs appear to be advancing slowly in a binary count pattern, which is driven by the simple_counter bits [26..23].

The LEDs are active low, therefore, when counting begins all LEDs are turned on (the 0000 state).

2. Press and hold KEY [0] on the development board and observe that the LEDs advance more quickly. Pressing this KEY causes the design to multiplex using the faster advancing part of the counter (bits [24..21]).

3. If other LEDs emit faintness light, select Assignments > Device. Click Device and Options. See Figure 6-49.

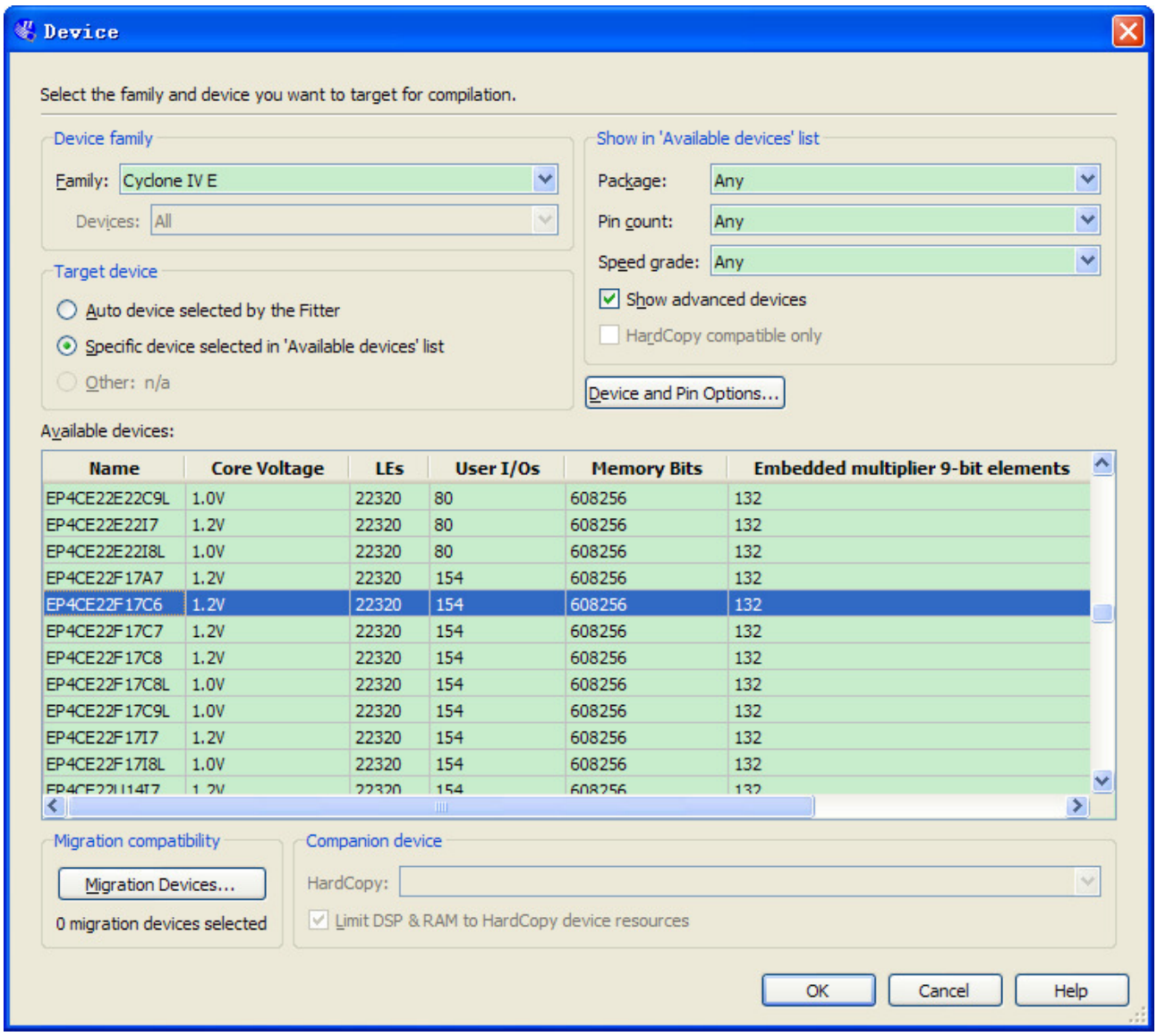

Figure 6-49 Device and Options

Select unused pins. Reserve all unused pins: select the **As input tri-stated** option. See Figure 6-50.

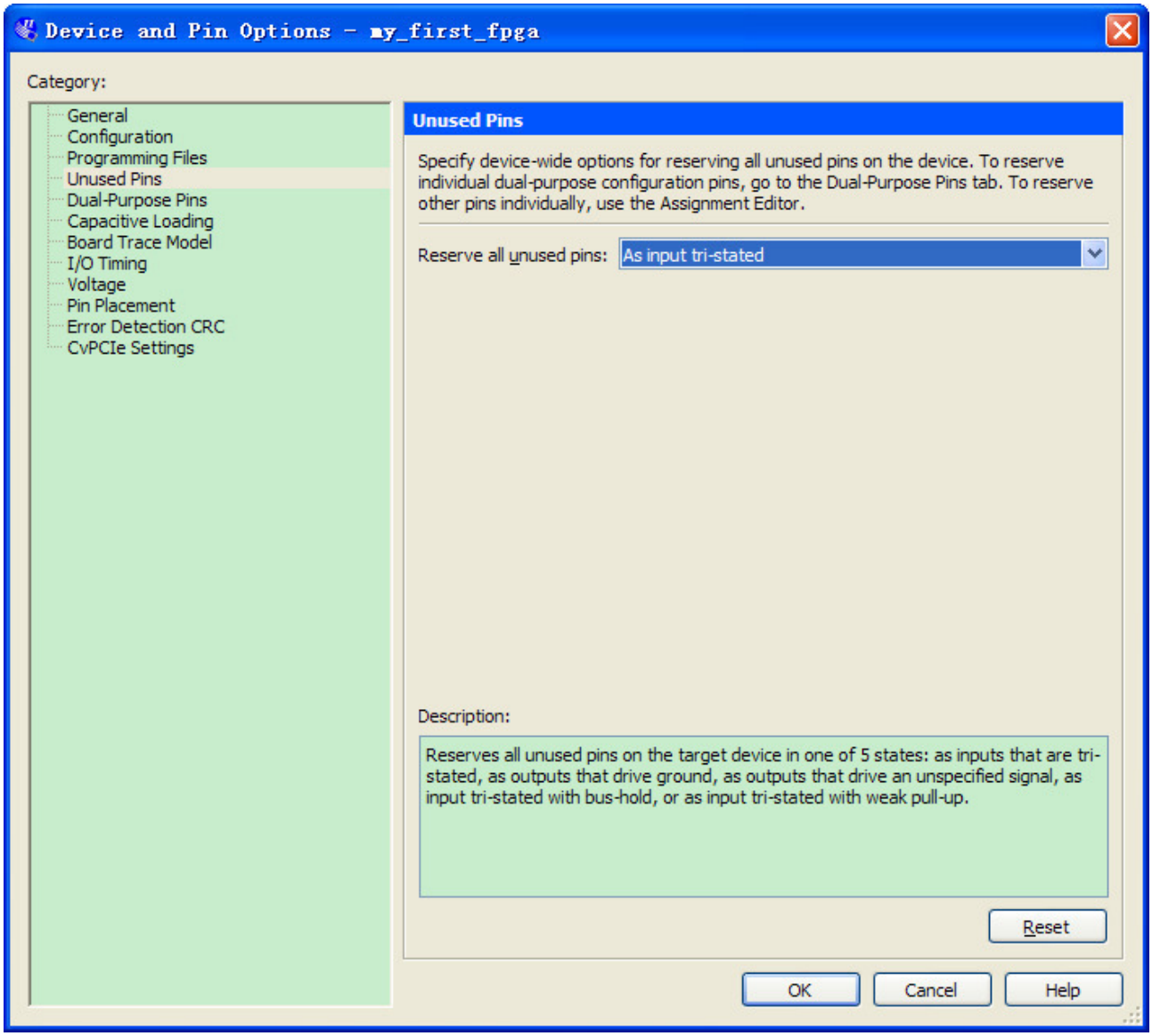

Figure 6-50 Setting unused pins

Click twice OK.

4. In the Processing menu, choose **Start Compilation**. After the compile, select **Tools > Programmer**. Select the **my_first_fpga.sof** file from the project directory. Click **Start**. At this time you could find the other LEDs are off.