



Copyright © 2003-2013 Terasic Technologies Inc. All Rights Reserved.





CHAPTER 1 INTRODUCTION	
1.1 Design Flow	1
1.2 Before You Begin	2
1.3 What You Will Learn	7
CHAPTER 2 ASSIGN THE DEVICE	
2.1 Assign The Device	
CHAPTER 3 DESIGN ENTRY	
3.1 Add a PLL Megafunction	
3.2 Add a Multiplexer	
3.3 Assign the Pins	
3.4 Create a Default TimeQuest SDC File	
CHAPTER 4 COMPILE AND VERIFY YOUR DESIGN	
4.1 Compile Your Design	
4.2 Program the FPGA Device	
4.3 Verify The Hardware	
CHAPTER 5 CONTACT US	
5.1 Headquarter & Branches	

## Chapter 1



This tutorial provides comprehensive information that will help you understand how to create a FPGA design and run it on you DE-Nano development board. The following sections provide a quick overview of the design flow, explain what you need to get started, and describe what you will learn.

## **1.1 Design Flow**

Figure 1-1shows the FPGA design flow block diagram.

The standard FPGA design flow starts with design entry using schematics or a hardware description language (HDL), such as Verilog HDL or VHDL. In this step, you can create a digital circuit that is implemented inside the FPGA. The flow then proceeds through compilation, simulation, programming, and verification in the FPGA hardware.

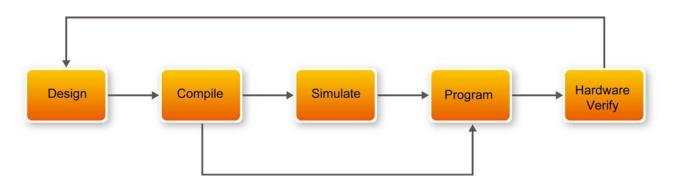


Figure 1-1 Design Flow

This tutorial guides you through all of the steps except for simulation. Although it is not covered in this document, simulation is very important to learn, and there are entire applications devoted to simulating hardware designs. There are two types of simulation, Functional and Timing Functional simulation allows you to verify that your code is manipulating the inputs and outputs appropriately. Timing (or post place-and-route) simulation verifies that the design meets timing and functions appropriately in the device.



### **1.2 Before You Begin**

This tutorial assumes the following prerequisites

■ You generally know what a FPGA is. 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 DE-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 you should installed the USB-Blaster driver. Use the USB cable to connect the leftmost USB connector on the DE-Nano board to a USB port on a computer that runs 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. The DE-Nano board is programmed by using Altera USB-Blaster mechanism. If the USB-Blaster driver is not already installed, the New Hardware Wizard in **Figure 1-2** will appear.







Figure 1-2 Found New Hardware Wizard

Since the desired driver is not available on the Windows Update Web site, select "No, not this time" in response to the question asked and click Next. This leads to the window in **Figure 1-3**.







Figure 1-3 The driver is found in a specific location

The driver is available within the Quartus II software. Hence, select Install from a specific location and click Next to get to Figure 1-4.



Found New Hardware Wizard
Please choose your search and installation options.
<ul> <li>Search for the best driver in these locations.</li> </ul>
Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.
Search removable media (floppy, CD-ROM)
Include this location in the search:
C:\altera\10.1\quartus\drivers\usb-blaster VBrowse
O Don't search. I will choose the driver to install.
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.
< Back Next > Cancel

Figure 1-4 Specify the location of the driver

Now, choose Search for the best driver in these locations and click Browse to get to the pop-up box in **Figure 1-5** Find the desired driver, which is at location C:\altera\10.1\quartus\drivers\usb-blaster. Click OK and then upon returning to **Figure 1-4** click Next. At this point the installation will commence, but a dialog box in **Figure 1-6** will appear indicating that the driver has not passed the Windows Logo testing. Click Continue Anyway.

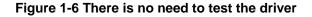




Browse For Folder	? 🗙
Select the folder that contains drivers for your hardware.	
🖃 🧰 10.1	~
🗉 🧰 installer	
🖽 🧰 ip	
🗄 🧰 nios2eds	
🖃 🧰 quartus	
🕀 🧰 bin	=
🗉 🧰 bin64	
🗷 🧰 common	
🗉 🧰 cusp	
🖃 🧰 drivers	
🚞 i386	
🗉 🧰 sentinel	
🖃 🗁 usb-blaster	
🗀 x32	
🚞 x64	
⊞	~
j Hill and a	
To view any subfolders, click a plus sign above.	
ОК	Cancel

Figure 1-5 Browse to find the location









The driver will now be installed as indicated in **Figure 1-7** Click Finish and you can start using the DE-Nano board.

Found New Hardware Wize	ard
	Completing the Found New Hardware Wizard The wizard has finished installing the software for: Altera USB-Blaster
	Click Finish to close the wizard.
	< Back Finish Cancel

Figure 1-7 The driver is installed

### 1.3 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 a speed that is controlled by an input key—This design is easy to create and gives you visual feedback that the design works. Of course, you can use your DE-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.

Develop a foundation to learn more about FPGAs—For example, you can create and download digital signal processing (DSP) functions onto a single chip, or build a multi-processor system, or create anything else you can imagine all on the same chip. You don't have to scour data books to find the perfect logic device or create your own ASIC. All you need is your computer, your imagination, and an Altera DE-Nano FPGA development board.



## Chapter 2

# Assign The Device

You 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.

### 2.1 Assign The Device

 In the Quartus II software, select File > New Project Wizard. The Introduction page opens. See Figure 2-1

🕊 New Project Vizard	
Introduction	
The New Project Wizard helps you create a new project and preliminary project settings, including the following: <ul> <li>Project name and directory</li> <li>Name of the top-level design entity</li> <li>Project files and libraries</li> <li>Target device family and device</li> <li>EDA tool settings</li> </ul> You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can u the various pages of the Settings dialog box to add functionality to the project.	ise
Don't show me this introduction again	
< Back Next > Finish Cancel Help	,

Figure 2-1 New Project Wizard introduction





- 2. Click Next.
- 3. Enter the following information about your project:
- 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.
- b. For example,  $E:My_design_my_first_fpga$ .
- c. File names, project names, and directories in the Quartus II software cannot contain spaces.
- d. What is the name of this project? Type my\_first\_fpga.
- e. What is the name of the top-level design entity for this project? Type my\_first\_fpga. See Figure 2-2.

Directory, Name, Top-Level Entity [page 1 of 5]         What is the working directory for this project?         E(Wy_design(my_frst_fpga	New Project Wizard	×
What is the working directory for this project?  [:[My_design/my_first_fpga What is the name of this project?  [my_first_fpga What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.  [my_first_fpga Use Existing Project Settings	Directory, Name, Top-Level Entity [page 1 of 5]	
E:/Wy_design/my_first_fpga What is the name of this project? my_first_fpga		
What is the name of this project?         my_first_fpga         What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.         my_first_fpga         (se Existing Project Settings)		
my_first_fpga          What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.         my_first_fpga          Wy_first_fpga          Use Existing Project Settings		
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.         my_first_fpga         Use Existing Project Settings		
my_first_fpga		
Use Existing Project Settings		
< Back Next > Finish Cancel Help	Use Existing Project Settings	
< Back Next > Finish Cancel Help		
<pre>&lt; Back Next &gt; Finish Cancel Help</pre>		
<pre><back next=""> Finish Cancel Help</back></pre>		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
< Back Next > Finish Cancel Help		
<pre>&lt; Back Next &gt; Finish Cancel Help</pre>		
	Sack Next > Finish Cancel H	telp

Figure 2-2 Project information

- f. Click Next.
- g. You will assign a specific FPGA device to the design and make pin assignments. See Figure 2-3.



					how in 'Availa	ble devices' list		
Eamily: Cyclone IV	/ F			▼ P		Any		~
					ac <u>k</u> age:	Any		
Dev <u>i</u> ces: All				Pi	Pin <u>c</u> ount: Any			~
Target device				s	Speed grade: Any			~
					Chow adva	anced devices		
<u>Auto device sele</u>	ected by the Fitter				Sliow auva	anced devices		
Specific device :	selected in 'Availabl	e devices'	list		Ha <u>r</u> dCopy	compatible only		
Other: n/a								
- Name	Core Voltage	LEs	User I/Os	Memory	Bits E	mbedded multiplier 9-bit elements	PLL	al d
	.2V	22320	80	608256	132	1	4	20
P4CE22E22I7 1		22320	80	608256	132	2	4	20
	.0V	22520						
P4CE22E22I8L 1	.0V .2V	22320	154	608256	132	1	4	20
P4CE22E22I8L 1 P4CE22F17A7 1			154 154	608256 608256	132 132		4 4	20 20
P4CE22E22I8L 1 P4CE22F17A7 1 P4CE22F17C6 1 P4CE22F17C7 1	.2V	22320	154 154					
P4CE22E22I8L         1           P4CE22F17A7         1           P4CE22F17C6         1           P4CE22F17C7         1           P4CE22F17C8         1	. 2V . 2V	22320 22320 22320 22320	154 154 154	608256 608256 608256	132 132 132		4	20 20 20
P4CE22E22I8L         1           P4CE22F17A7         1           P4CE22F17C6         1           P4CE22F17C7         1           P4CE22F17C8         1           P4CE22F17C8         1	2V 2V 2V 2V 0V	22320 22320 22320 22320 22320 22320	154 154 154 154	608256 608256 608256 608256	132 132 132 132		4	20 20 20 20
P4CE22E22I8L 1 P4CE22F17A7 1 P4CE22F17C6 1 P4CE22F17C6 1 P4CE22F17C7 1 P4CE22F17C8 1 P4CE22F17C8L 1	2V 2V 2V 2V	22320 22320 22320 22320	154 154 154 154 154	608256 608256 608256 608256 608256	132 132 132		4 4 4	20 20 20 20
P4CE22E22I8L 1 P4CE22F17A7 1 P4CE22F17C6 1 P4CE22F17C6 1 P4CE22F17C7 1 P4CE22F17C8 1 P4CE22F17C8L 1	2V 2V 2V 2V 0V	22320 22320 22320 22320 22320 22320	154 154 154 154	608256 608256 608256 608256 608256	132 132 132 132		4 4 4	20 20 20 20
P4CE22E22I8L 1 P4CE22F17A7 1 P4CE22F17C6 1 P4CE22F17C7 1 P4CE22F17C8 1 P4CE22F17C8L 1	2V 2V 2V 2V 0V	22320 22320 22320 22320 22320 22320	154 154 154 154 154	608256 608256 608256 608256 608256	132 132 132 132		4 4 4	20 20 20 20
P4CE22E22I8L 1 P4CE22F17A7 1 P4CE22F17C6 1 P4CE22F17C6 1 P4CE22F17C7 1 P4CE22F17C8 1 P4CE22F17C8L 1	2V 2V 2V 2V 0V	22320 22320 22320 22320 22320 22320	154 154 154 154 154	608256 608256 608256 608256 608256	132 132 132 132		4 4 4	20 20 20 20

Figure 2-3 Specify the Device Example

h. Click Finish.

terasic

4. When prompted, choose Yes to create the my\_first\_fpga project directory. You just created your first Quartus II FPGA project. See Figure 2-4.



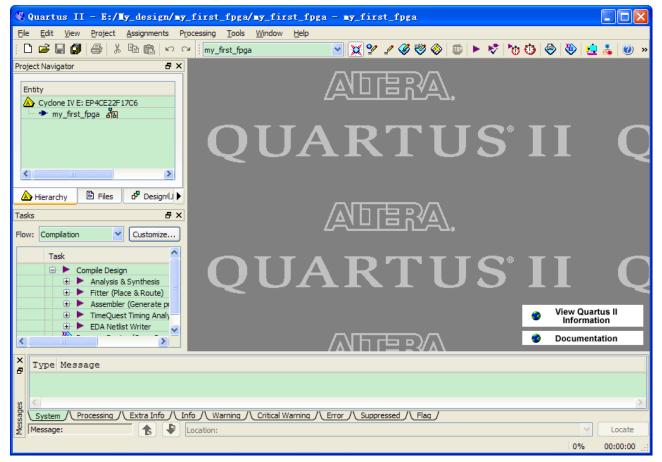


Figure 2-4 my\_first\_fpga project



# Chapter 3 Design Entry

### **3.1 Add a PLL Megafunction**

This section describes How to Add a PLL Megafunction

In the design entry step you create a schematic or Block Design File (.bdf) that is the top-level design. You will add library of parameterized modules (LPM) functions and use Verilog HDL code to add a logic block. When creating your own designs, you can choose any of these methods or a combination of them.

1. Choose File > New > Block Diagram/Schematic File (see Figure 3-1 to create a new file, Block1.bdf, which you will save as the top-level design.

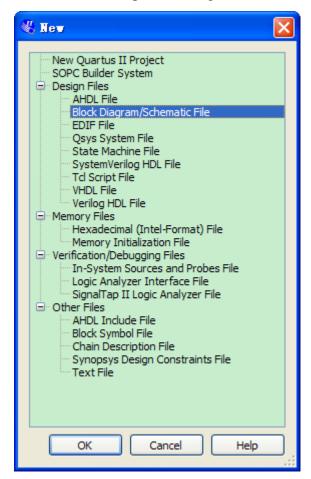


Figure 3-1 New BDF





- 2. Click OK.
- 3. Choose 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 3-2).

🖑 Quartus II - E:/Ey_design/my	y_first_fpga/my_first_fpga - my_first_fpga	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>A</u> ssignments P	P <u>r</u> ocessing <u>T</u> ools <u>W</u> indow <u>H</u> elp	
i 🗋 🖨 🔚 🎒 🎒 👗 🖻 🖀 👘	🗠 i my_first_fpga 💦 😵 😰 🖉 🏈 🐨 🕨 🤣 🕹 😓 🌡	🛓 🕜 »
Project Navigator 🛛 🗗 🗙		
		🚑   »
Entity		
A Cyclone IV E: EP4CE22F17C6 → my_first_fpga 品		
		:::::
A Hierarchy  ☐ Files		
		::::: =
Tasks & 🗗 🗙	•••••••••••••••••••••••••••••••••••••••	
Flow: Compilation V Customize		
lask		
Compile Design		
Analysis & Synthesis      Fitter (Place & Route)		
Assembler (Generate pr	•••••••••••••••••••••••••••••••••••••••	:::::=
🕀 🕨 TimeQuest Timing Analy		
🕀 🕨 EDA Netlist Writer 🗸		
	<	
X Turnel Manager		
× Type Message		
s <		5
System / Processing / Extra Info / I	Info /\ Warning /\ Critical Warning /\ Error /\ Suppressed /\ Flag /	
Message:	Location:	Locate
	638, 218 0% 0	0:00:00

Figure 3-2 Bank BDF

- 5. Add HDL code to the blank block diagram by choosing File > New > Verilog HDL File.
- 6. Click OK to create a new file Verilog1.v, which you will save as simple\_counter.v.
- 7. Select File > Save As and enter the following information (see Figure 3-3).

File name: simple\_counter.v

Save as type: Verilog HDL File (\*.v, \*.vlg, \*.verilog)





Save As						? 🗙
Save in: My Recent Documents Desktop My Documents My Computer	<ul> <li>my_first_fpga</li> <li>db</li> <li>greybox_tmp</li> <li>incremental_db</li> <li>counter_bus_m</li> <li>counter_bus_m</li> <li>pll_v</li> <li>pll_bb.v</li> <li>simple_counter</li> </ul>	ux.v ux_bb.v	• •		*	
My Network Places	File name: Save as type:	simple_counter.v Verilog HDL Files (*.v *.vlg *.v Add file to current project	erilog)	•	•	Save Cancel

Figure 3-3 Saving the Verilog HDL file

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

- 8. Type the following Verilog HDL code into the blank simple\_counter.v file (see **Figure 3-4** The Verilog File of simple\_counter.v).
- //It has a single clock input and a 32-bit output port

module simple\_counter (

CLOCK\_5,

counter\_out

);

input CLOCK\_5;

ter asiC

output [31:0] counter\_out;

reg [31:0] counter\_out;



always @ (posedge CLOCK\_50) // on positive clock edge begin

```
counter_out <= counter_out + 1;</pre>
```

// 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
                        CLOCK 5 ;
      input
               [31:0] counter_out;
 7
      output
8
               [31:0] counter_out;
      rea
9
10
      always @ (posedge CLOCK 5)
                                               // on positive clock edge
         begin
11
    Ξ
12
            counter out <= counter out + 1;
                                              // increment counter
13
         end
14
      endmodule
                                                // end of module counter
15
```

#### Figure 3-4 The Verilog File of simple\_counter.v

- 9. Save the file by choosing File > Save, pressing Ctrl + S, or by clicking the floppy disk icon.
- 10. Choose File > Create/Update > Create Symbol Files for Current File to convert the simple\_counter.v file to a Symbol File (.sym).You use this Symbol File to add the HDL code to your BDF schematic.

The Quartus II software creates a Symbol File and displays a message (see Figure 3-5).

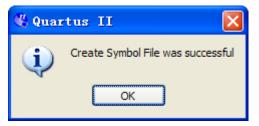


Figure 3-5 Create Symbol File was Successful

- 11. Click OK.
- 12. To add the simple\_counter.v symbol to the top-level design, click the my\_first\_fpga.bdf tab.



ter <mark>asIC</mark>

- 13. Choose Edit > Insert Symbol.
- 14. Double-click the Project directory to expand it.
- 15. Select the newly created simple\_counter symbol by clicking it's icon.

You can also double-click in a blank area of the BDF to open the Symbol dialog box (If your Quartus II version is lower than 10.0)

a Symbol	E
Libraries:	
<ul> <li> <b>Project</b> </li> <li> <b>d</b>:/altera/10.1/quartus/libraries/      </li> </ul>	simple_counter CLOCK_5 counter_out[31.0]
Name:	inst
simple_counter	
Insert symbol as block	
Launch MegaWizard Plug-In	
MegaWizard Plug-In Manager	
	OK Cancel

Figure 3-6 Adding the Symbol to the BDF

- 16. Click OK.
- 17. 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 3-7**.



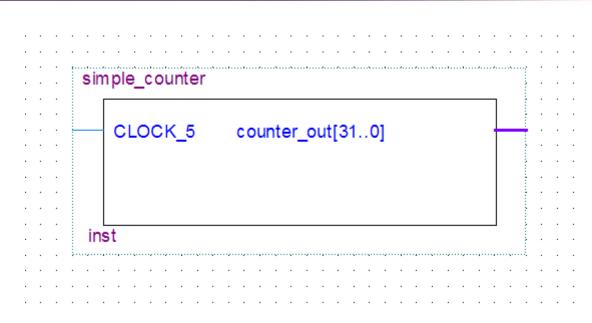


Figure 3-7 Placing the simple\_counter symbol

- 18. Press the Esc key or click an empty place on the schematic grid to cancel placing further instances of this symbol.
- 19. Save your project regularly.

Using Quartus Add a PLL Megafunction

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 (DE-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. Choose Edit > Insert Symbol or click Add Symbol on the toolbar
- Click Megawizard Plug-in Manager. The MegaWizard® Plug-In Manager appears (see Figure 3-8).





🐇 IegaViza	ard Plug-In Manager [page 1]	×
	The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions. Which action do you want to perform? © <u>Greate a new custom megafunction variation</u> © Edit an existing custom megafunction variation © Cogy an existing custom megafunction variation Copyright (C) 1991-2010 Altera Corporation	
	Cancel     < Back     Next >     Finish	

Figure 3-8 Mega Wizard Plug-In Manager

- 3. Click Next.
- 4. In MegaWizard Plug-In Manager [page 2a], specify the following selections (see **Figure 3-9**):
- a. Choose I/O > ALTPLL.
- b. Under Which device family will you be using? Choose the Cyclone IV E for DE-Nano development board.
- c. Under Which type of output file do you want to create? Choose 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.





🖑 MegaVizard Plug-In Manager [pa	ge 2a]
Which megafunction would you like to customize? Select a megafunction from the list below          ALTDDIO_IN         ALTDDIO_OUT         ALTDQ         ALTDQ         ALTDQ DQS         ALTDQ_DQS         ALTDQ_DQS         ALTGX_RECONFIG         ALTOBUF         ALTOBUF	ge 2a]       X         Which device family will you be using?       Cyclone IV E         Which type of output file do you want to create?       X         AHDL       YHDL         YHDL       Verilog HDL         What name do you want for the gutput file?          E:/My_design/my_first_fpga/pll.v
ALTEVOS_CK ALTEVOS ALTEVO	Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in a library specified in the Libraries page of the Options dialog box (Tools menu), or a library specified in the Libraries page of the Settings dialog box (Assignments menu). Your current user library directories are:
MAX II/MAX V oscillator	Cancel  < Back

Figure 3-9 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 3-10).
- a. Confirm that the Currently selected device family option shows the device family that corresponds to the development board you are using.
- b. The device speed grade choose 6 for DE-Nano.
- c. Set the frequency of the inclock0 input 50 MHz.
- d. Click Next.



### ter <mark>asIC</mark>

🔨 MegaVizard Plug-In Manager [pag	ze 3 of 14] ? 🔀
ALTPLL	About Documentation
Parameter         2 PLL         3 Output           Settings         Reconfiguration         Clocks	Image: Application of the second s
General/Modes Inputs/Lock Bandwid	Currently selected device family: Cydone IV E Match project/default
	Cancel < Back Next > Einish

Figure 3-10 MegaWizard Plug-In Manager [page 3 of 14] Selections

6. Turn off all options on MegaWizard page 4. As you turn them off, pins disappear from the PLL block's graphical preview. See **Figure 3-11** for an example.



≺ TegaVizard Plug-In Tanager [pag	e 4 of 14] 🤶 🔀
ALTPLL	<u>About</u> <u>Documentation</u>
Parameter Settings         2 PLL Reconfiguration         3 Output Clocks	4 EDA 5 Summary
General/Modes Inputs/Lock Bandwidt	h/SS > Clock switchover >
Image: pli field frequency: 50.000 MHz         Operation Mode: Normal         Image: pli field frequency: 50.000 MHz         Operation Mode: Normal         Image: pli field frequency: field freq	Able to implement the requested PLL         Optional Inputs <ul> <li>Greate an 'pliena' input to selectively enable the PLL</li> <li>Greate an 'pfiena' input to selectively enable the plase/frequency detector</li> </ul> Lock Output <ul> <li>Greate an 'pfiena' input to selectively enable the phase/frequency detector</li> </ul> Lock Output <ul> <li>Greate on loss lock</li> </ul> Advanced Parameters <ul> <li>Using these parameters is recommended for advanced users only</li> <li>Greate output file(s) using the 'Advanced' PLL parameters</li> <li>Configurations with output clock(s) that use cascade counters are not supported</li> </ul>
	Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

Figure 3-11 MegaWizard Plug-In Manager [page 4 of 14] Selections

- 7. Click Next four times.
- 8. At the top of the wizard, click the tab 3. Output Clocks to jump to the Output Clocks > clk c0 page

Clock Division Settings () input 10 (Figure 3-12).





🔨 MegaVizard Plug-In Manager	page 6 of 12]			? 🛛
				About Documentation
URSER!				
1 Parameter 2 PLL 3 Out Settings Reconfiguration Close	ut 4 EDA 5 Summ	ary		
dkc0 $dkc1$ $dkc2$ $dkc2$				
pli	c0 - Core/Ex Able to implement	ternal Output Cle	ock	
jnclk0	Able to implement i			
inclk0 frequency: 50.000 MHz Operation Mode: Normal	Use this clock			
Clk Ratio Ph (dg) DC (%)	Clock Tap Setti	ngs		
c0 1/10 0.00 50.00			Requested Settings	Actual Settings
Cyclone IV E	,	tput clock frequency:	100.0000000	MHz 🕑 5.000000
	<ul> <li>Enter ou</li> </ul>	tput clock parameters:		
	Clock mu	ltiplication factor	1	1
	Clock div	ision factor	10 🗘	<< Copy 10
	Clock phase	shift	0.00	deg 💙 0.00
	Challed at the second		50.00	50.00
	Clock duty cy	/cle (%)	50.00	50.00
			Description	Value
			Primary clock VCO frequency (MHz)	600.000
		splayed internal	Modulus for M counter	12
		ne PLL is recommended dvanced users only	Modulus for N counter Initial VCO phase cycles for M count	1 er 1
			VCO phase tap for M counter	0
			-Per Cloc	k Feasibility Indicators
			c0	c1 c2 c3 c4
			Cancel < E	Back Next > Finish
			Cancer < t	

Figure 3-12 MegaWizard Plug-In Manager [page 8 of 14] Selections

- 9. Click Finish.
- 10. The wizard displays a summary of the files it creates (see **Figure 3-13**). Select the pll.bsf option and click Finish again.

### ter asIC

🔨 MegaVizard Plug-In Manager [pag	e 14 of 14]	? 🛛
		About Documentation
Parameter         PLL         3 Output           Settings         Reconfiguration         Clocks	4 EDA 5 Summary	
pl         inclk0       inclk0 frequency: 50.000 MHz         Operation Mode: Normal         © 1/1       0.00         Cyclone IV E	checkmark indicates an optic maintained in subsequent M	Description Variation file PinPlanner ports PPF file AHDL Include file VHDL component declaration file Quartus II symbol file Instantiation template file Verilog HDL black-box file
		Cancel <u>Back</u> <u>Next&gt;</u>

Figure 3-13 Wizard-Created Files

The Symbol window opens, showing the newly created PLL megafunction. See Figure 3-14.



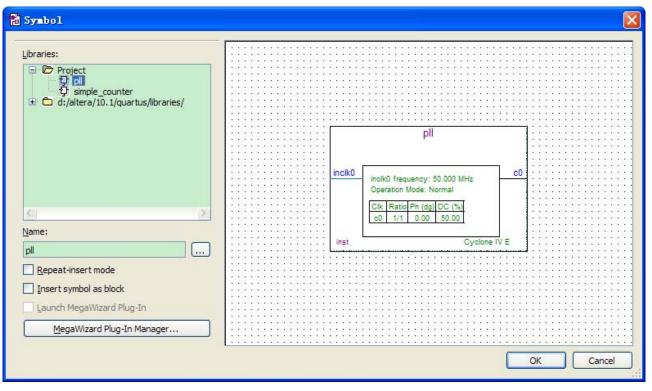
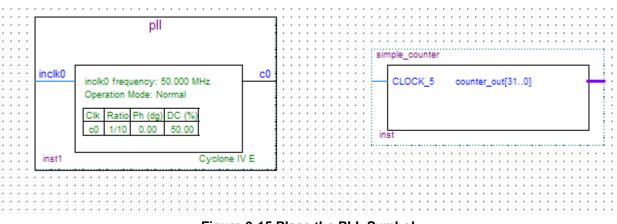


Figure 3-14 PLL Symbol

11. Click OK and place the pll symbol onto the BDF to the left of the simple\_counter symbol. You can move the symbols around by holding down the left mouse button, helping you ensure that they line up properly. See **Figure 3-15**.



- Figure 3-15 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 3-16).

ter <mark>asIC</mark>

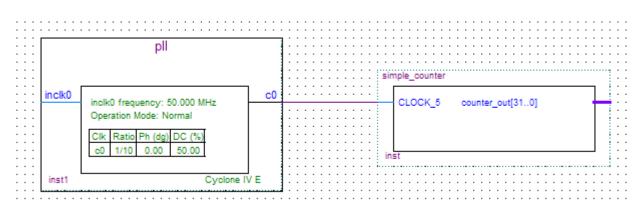


Figure 3-16 Draw a Bus Line connect pll c0 port to simple\_counter CLOCK\_5 port

- 14. Add an input pin and an output bus with the following steps:
- a. Choose Edit > Insert Symbol.
- b. Under Libraries, select quartus/libraries > primitives > pin >input. See Figure 3-17
- c. 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.

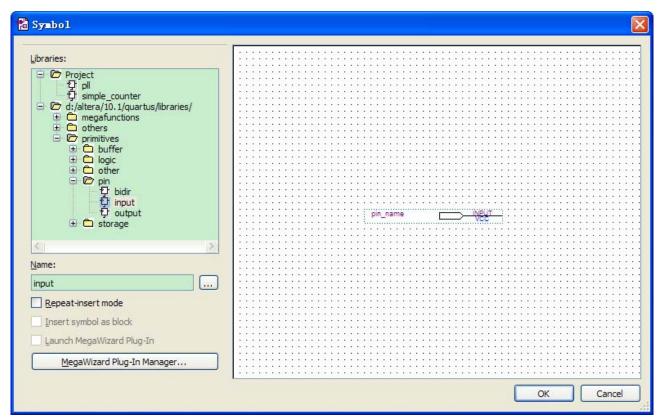


Figure 3-17 Input pin symbol

- d. Place the new pin onto the BDF so that it is touching the input to the pll symbol.
- e. Use the mouse to click and drag the new input pin to the left; notice that the ports remain



connected as shown in Figure 3-18.

pli			· · · · · · · ·	· · · ·		::										
inclk0 frequency: 5 VCC VCC Clk Ratio Ph (dg) C0 1/10 0.00		· · · · · · · · · · · · · · · · · · ·	· · · ·	· · · ·	· · · · · · · · · · ·	inst	CLOC	K_5	c	ounte	er_ou	ıt[31.	.0]			•••••••••••••••••••••••••••••••••••••••
inst1	Cyclone IV E		· · · ·		· · · · · · · · · · · ·	· · · ·	· · ·	· · · ·	· · · ·					 • •	· · · ·	 •

Figure 3-18 Connecting the PLL symbol and Input port

- f. Change the pin name by double-clicking pin\_name and typing CLOCK\_50 (see **Figure 3-19**). This name correlates to the oscillator clock that is connected to the FPGA.
- g. 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.

📸 Pin Propert	ties	×
General Form	at	_
To create multip (For example: "r	le pins, enter a name in AHDL bus notation name[30]"), or enter a comma-seperated list of names.	_
<u>P</u> in name(s):	CLOCK_50	
Default value:	vcc	
	OK Cancel Help	

Figure 3-19 Change the input port name

h. Right-click the new output bus line and choose Properties.

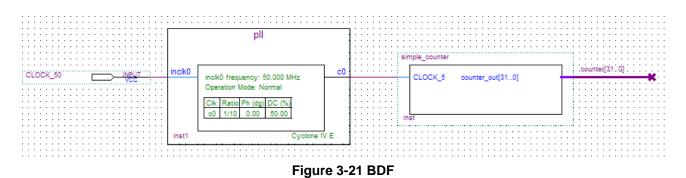




- i. Using the Orthogonal Bus tool, draw a bus line connected on one side to the simple\_counter output port, and leave the other end unconnected at about 6 to 8 grid spaces to the right of the simple\_counter.
- j. Type counter [31..0] as the bus name (see **Figure 3-20**). The notation [X ..Y] is the 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).
- k. Click OK. Figure 3-21 shows the BDF.

Bus Properties	×
General Font Format	
Name: counter[310]	
Hide name in block design file.	
OK Cancel Help	

#### Figure 3-20 Change the output BUS name



### 3.2 Add a Multiplexer

This design uses a multiplexer to route the simple\_counter output to the LED pins on the DE-Nano development board. You will use the MegaWizard Plug-In Manager to add the multiplexer, lpm\_mux. The design multiplexes two variations of the counter bus to four LEDs on the DE-Nano development board.

- 1. Choose Edit > Insert Symbol.
- 2. Click Megawizard Plug-in Manager.
- 3. Click Next.
- 4. Choose Installed Plug-Ins > Gates > LPM\_MUX.
- 5. Choose the device family that corresponds to the device on the development board you are using, choose Verilog HDL as the output file type, and name the output file counter\_bus\_mux.v (see Figure 3-22).
- 6. Click Next.

🐇 MegaVizard Plug-In Manager [pag	ge 2a]	Σ	K
Which megafunction would you like to customize? Select a megafunction from the list below Installed Plug-Ins Altera SOPC Builder Arithmetic Communications DSP Gates LPM_CLSHIFT LPM_CONSTANT LPM_DECODE LPM_MUX I/O Interfaces JTAG-accessible Extensions Memory Compiler	Which device family will you be using? Which type of output file do you want to AHDL YHDL Verilog HDL What name do you want for the gutput file E:/My_design/my_first_fpga/counter_bile Return to this page for another creat Note: To compile a project successfully in files must be in the project directory, in a the Options dialog box (Tools menu), or of the Settings dialog box (Assignments of Your current user library directories are:	file? us_mux.v te operation n the Quartus II software, your design a library specified in the Libraries page of a library specified in the Libraries page menu).	
	Cancel	Back Next > Finish	

Figure 3-22 Selecting Ipm\_mux

terasic

ter <mark>asIC</mark>

- 7. Under How many 'data' inputs do you want? Select 2 inputs (default).
- 8. Under How 'wide' should the data input and result output be? Select 4 (see Figure 3-23).

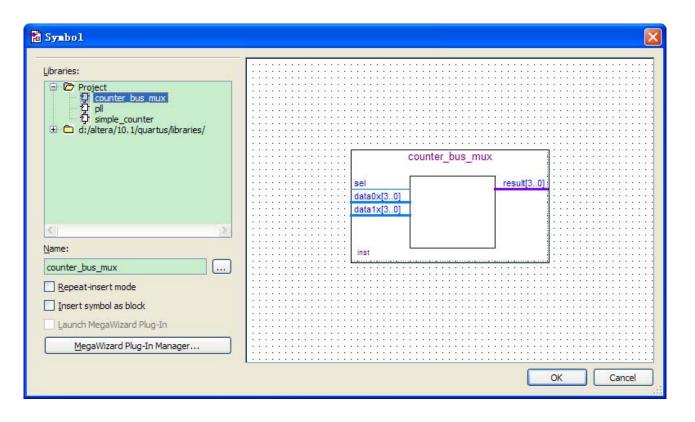
▲ MegaVizard Plug-In Manager [page]	e 3 of 5] 🔹 🥐 🔀
DPM_MUX	<u>About</u> Documentation
1 Parameter Settings 2 EDA 3 Summary	
counter_bus_mux         sel	Currently selected device family:     Cyclone IV E     Match project/default     How many 'data' inputs do you want?     Image: Comparison of the second sec
1 lpm_mux	Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

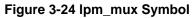
Figure 3-23 lpm\_mux settings

- 9. Click Next.
- 10. Click Next, select the counter\_bus\_mux.bsf options.
- 11. Click Finish. The Symbol window appears (see Figure 3-24 for an example).









#### 12. Click OK

13. Place the counter\_bus\_mux symbol below the existing symbols on the BDF. See Figure 3-25.

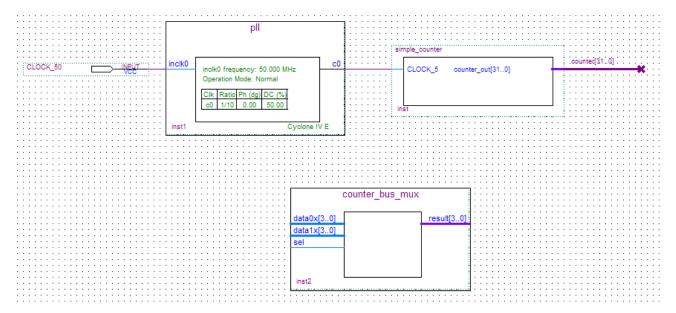


Figure 3-25 Place the lpm\_mux symbol

- 14. 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.

### ter <mark>asIC</mark>

- b. Draw a bus line from the result [3..0] output port to about 4 to 8 grid spaces to the right of counter\_bus\_mux.
- c. Right-click the bus line connected to data1x[3..0] and choose 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 choose 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 3-26 shows the renamed buses.

· · · · · · · · · · · · · · · ·				• • •	• •	•	• •	• •		•
	counter_bus_mux			• •	• •	•	• •	• •	•	•
				• • •	• •	•	• •	• •	-	•
				• •	• •	•	• •	• •		•
.counter[2421]	data0x[30] re	esult[30]	re	sult	[30	<u>)</u> ]	: :			:
	dataox[00]	Souriool :		_		-	_	-	-	-1
.counter[2623]	data1x[30]			• • •		•	• •	• •	•	÷
				• • •	• •	• •	• •	• •	-	•
	sel			• • •	• •	•	• •	• •	-	•
				• • •	• •	•	• •	• •	•	•
				• • •	• •	•	• •	• •	-	•
				• • •	• •	•	• •	• •		•
			· · ·		• •	•	• •	• •	•	•
				• • •	• •	•	• •	• •	•	•
	in a t 2			• • •	• •	•	• •	• •	•	•
	inst2			• • •			• •		-	•

Figure 3-26 Renamed counter\_bus\_mux Bus Lines

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

- 15. Choose Edit > Insert Symbol.
- 16. Under Libraries, double-click quartus/libraries/ > primitives > pin > output (see Figure 3-27).



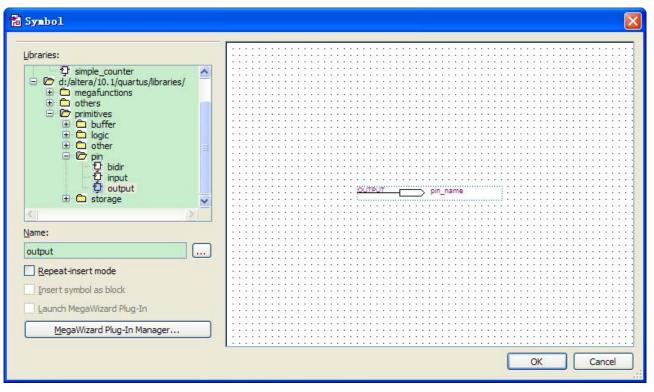
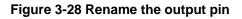


Figure 3-27 choose an output pin

- 17. Click OK.
- 18. Place this output pin so that it connects to the counter\_bus\_mux result [3..0] bus output line.
- 19. Rename the output pin as LED [3..0] as described in steps 13 c and d. (see Figure 3-28).

						 	• •				· ·	 		
· · · · · · · · · · · · · · · · · · ·		counter_bus_mux			· · · ·	 · · ·	· · · ·	· · · ·	· · ·	· · ·		  · · · · · · ·	· · · ·	 
.counter[2421]	data0x[30]		result[30]			 						 LED		
	data1x[30]													
••••••••••	sel			I										
				I										
					· ·									
	inst2			I										
				<b>1</b> · · ·	: :		: :	: :				 	· ·	



- 20. Attach an input pin to the multiplexer select line using an input pin:
- a. Choose Edit > Insert Symbol.
- b. Under Libraries, double-click quartus/libraries/ > primitives > pin > input.
- c. Click OK.
- 21. Place this input pin below counter\_bus\_mux.
- 22. Connect the input pin to the counter\_bus\_mux sel pin.





23. Rename the input pin as KEY [0] (see Figure 3-29).

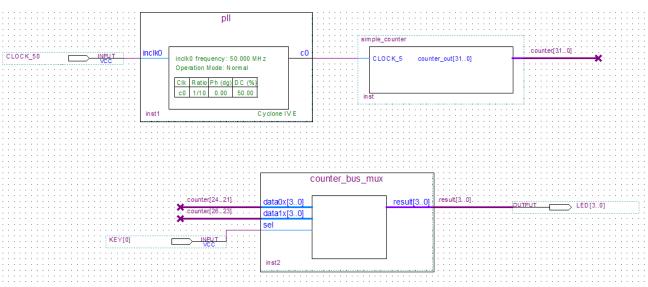


Figure 3-29 Adding the KEY [0] Input Pin

You have finished adding symbols 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 "My First FPGA Project."

### **3.3 Assign the Pins**

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

- 1. Choose 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 that correlate to the KEY [0] and CLOCK\_50 input pins and LED[3..0] output pin, perform the following steps:

1. Choose Assignments > Pins, which opens the Pin Planner, a spreadsheet-like table of specific pin assignments. The Pin Planner shows the design's six pins. See Figure 3-30



Groups     To View We But       Named: *        Node Name     Direction       Location        KEY[00]     Input Group       Convert        KEY[00]     Output Group       Convert        Named: *					5 ×		ups	Gro
Named: *  Named: * Na							·	
<ul></ul>							ned: * 💉 💙	Nan
Imput Group         Input Group           Imput Group         Output Group           Imput Group         Imput Group           Imput Group					Location	Direction	Node Name	
Image: Ward State     Image: State       Image: State     Image: State       Image: State     Image: State		XAXAXAAAAA				Input Group	WEY[00]	÷
Named: *         ★			0000			Output Group	LED[30]	۲
Named: * Kalt: X V							< <new group="">&gt;</new>	- L.,
Named: * • K - Edit: * -			Aco					
		<u> </u>						
C Node Name Direction Location I/O Bank VREE Group I/O Standard	Filter: Pins: all					»Edit: 🗙 🗸	Named: *	
Note Name Direction Eccadori 1/0 Bank Vice Group 1/0 Standard	Reserved	I/O Standard	VREF Group	I/O Bank	Location	Direction	Node Name	P
CLOCK_50 Input 2.5 V (default)		2.5 V (default)				Input	CLOCK_50	
Imp KEY[0] Input 2.5 V (default)		2.5 V (default)				Input	KEY[0]	
Output     2.5 V (default)		2.5 V (default)				Output	LED[3]	
@ LED[2] Output 2.5 V (default)						Output		
						Output		
		2.5 V (default)				Output		
							< <new node="">&gt;</new>	
IED[0]          Output         2.5 V (default)           < <new node="">&gt;</new>								
Image: Constraint of the state of		2.5 V (default) 2.5 V (default)				Output Output	<ul> <li>LED[2]</li> <li>LED[1]</li> <li>LED[0]</li> </ul>	

Figure 3-30 Pin Planner Example

1. In the Location column next to each of the six node names, add the coordinates (pin numbers) as shown in **Table 3-1** for the actual values to use with your DE-Nano board.

	8
Pin Name	FPGA Pin Location
KEY[0]	J15
LED[3]	A11
LED[2]	B13
LED [1]	A13
LED [0]	A15
CLOCK_50	R8

#### **Table 3-1 Pin Information Setting**

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. See **Figure 3-31**.

erias (

	<u> </u>	dit <u>V</u> iew P <u>r</u> ocessing	<u>T</u> ools <u>W</u> indow						
₹	Gro	ups		₽×			Top View - Wire Bond Ione IV E - EPt0E0051706		
~	Nam	ned: * 💉 🖌							
3		Node Name	Direction	Locati 🛆		- 000			
_		🗆 💿 LED[2]	Output	PIN_B13					
b3		🕑 LED[1]	Output	PIN_A13					
Ð		💷 💿 LED[0]	Output	PIN_A15					
3	<	< <new group="">&gt;</new>							
e,	×	Named: * 🗸 🕅	»Edit: 🗙 🗸		Filter: Pins: all				•
n	8	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
ca		CLOCK_50	Input	PIN_R8	3	B3_N0	2.5 V (default)		
_		KEY[0]	Input	PIN_J15	5	B5_N0	2.5 V (default)		
		LED[3]	Output	PIN_A11	7	B7_N0	2.5 V (default)		
Y		LED[2]	Output	PIN_B13	7	B7_N0	2.5 V (default)		
E		LED[1]	Output	PIN_A13	7	B7_N0	2.5 V (default)		
		LED[0]	Output	PIN_A15	7	B7_N0	2.5 V (default)		
		< <new node="">&gt;</new>							
<b>11</b> 2	All Pins								

Figure 3-31 Completed Pin Planning Example

Now, you are finished creating your Quartus II design!

### 3.4 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. Choose 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

er as (

derive\_clock\_uncertainty

4. Save this file as my\_first\_fpga.sdc (see Figure 3-32)



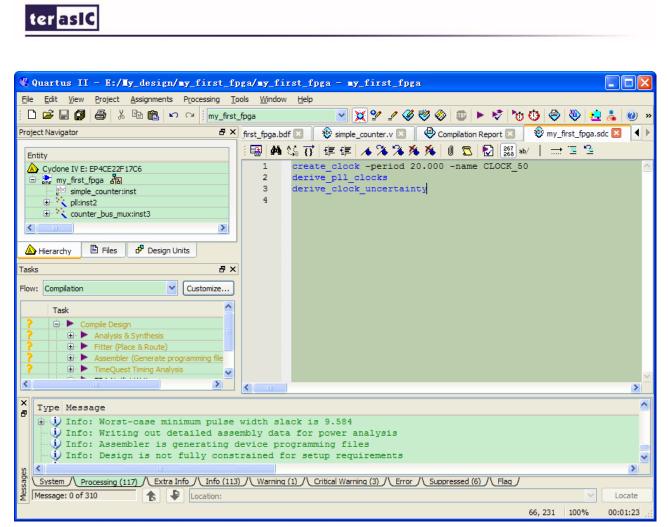


Figure 3-32 Default SDC

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



### Chapter 4

# Compile and Verify 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. The software also generates other report files that provide information about your code as it compiles.

### **4.1 Compile Your Design**

If you want to store .SOF in memory device (such as flash or EEPROMs), you must first convert the SOF to a file type specifically for the targeted memory device.

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

In the Processing menu, choose 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 (see Figure 4-1).

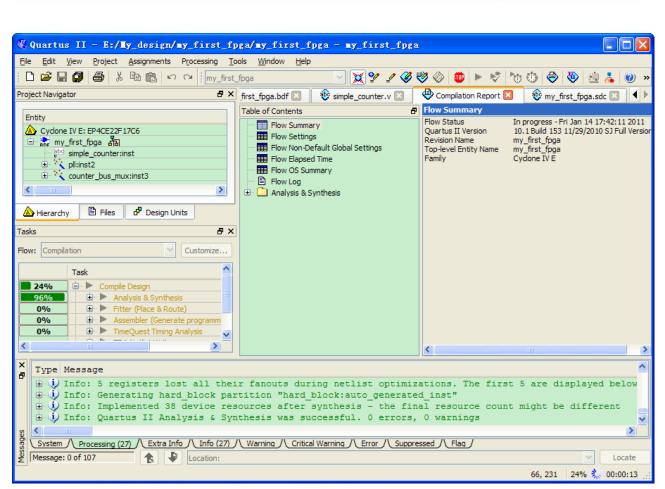


Figure 4-1 Compilation Message for project

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 4-2**.

terasic



Flow Summary						
Flow Status Quartus II Version Revision Name Top-level Entity Name Family Device Timing Models Total logic elements Total combinational functions Dedicated logic registers Total registers Total pins Total pins Total pins Total wirtual pins Total memory bits Embedded Multiplier 9-bit elements Total PLLs	Successful - Fri Jan 14 17:42:39 2011 10.1 Build 153 11/29/2010 SJ Full Version my_first_fpga Cyclone IV E EP4CE22F17C6 Final 31 / 22,320 ( < 1 % ) 27 / 22,320 ( < 1 % ) 27 6 / 154 ( 4 % ) 0 0 / 608,256 ( 0 % ) 0 / 132 ( 0 % ) 1 / 4 ( 25 % )					

Figure 4-2 Compilation Report Example

### 4.2 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:

For the DE-Nano board, connect the USB-Blaster (included in your development kit) to J3 and the USB cable to the USB-Blaster. Connect the other end of the USB cable to the host computer.

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

Program the FPGA using the following steps.

1. Choose Tools > Programmer. The Programmer window opens. See Figure 4-3.



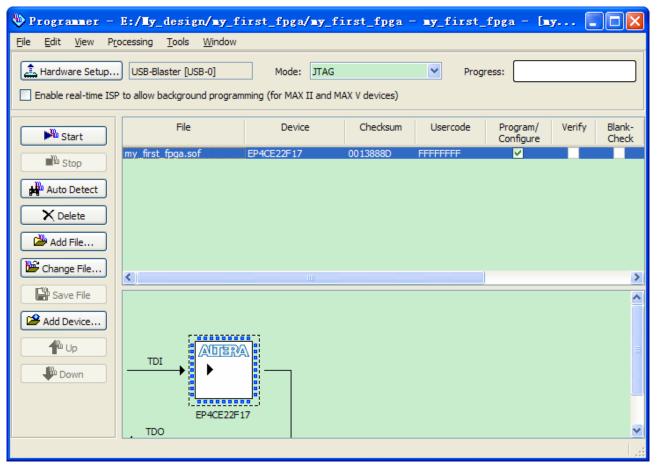


Figure 4-3 Programmer Window

- 2. Click Hardware Setup.
- 3. If it is not already turned on, turn on the USB-Blaster [USB-0] option under currently selected hardware. See Figure 4-4.



Ð	Hardware Setup			X			
	Hardware Settings JTAG Se Select a programming hardware hardware setup applies only to	setup to use when progra		'his programming			
	Currently selected hardware:	USB-Blaster [USB-0] No Hardware USB-Blaster [USB-0]					
	Hardware	Server	Add Hardware				
	USB-Blaster	Local	USB-0	Remove Hardware			
Close							

#### Figure 4-4 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 4-5).



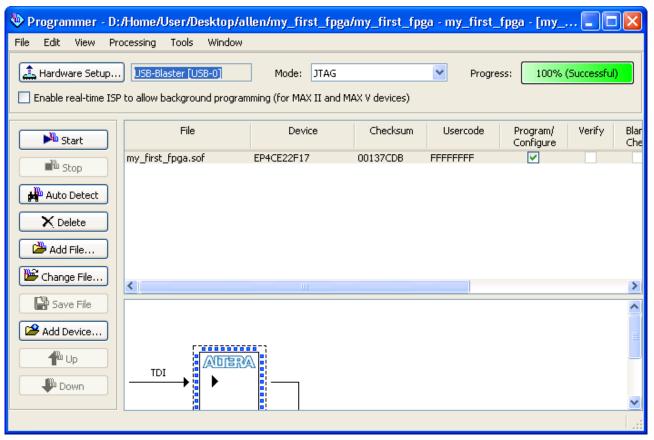


Figure 4-5 Downloading Complete

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.

### 4.3 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]).



ter <mark>asIC</mark>

3. If other LEDs emit faintness light, Choose Assignments > Device. Click Device and Options. See Figure 4-6.

Transmission of the second sec				Show in 'Available devices' list			
Eamily: Cyclone	IVE		~	Package:	Any	~	
			trane.				
Devices: All			~	Pin <u>c</u> ount:	Any	~	
Target device				Speed grade:	Any	~	
and the second second				Show advanced devices			
O Auto device :	selected by the Fitter			HardCopy compatible only			
Specific device	ce selected in 'Availab	le devices	list	- Harocopy (	compauble only		
O Other: n/a				De la sel Die d			
				Device and Pin C	Options		
<u>vailable</u> devices:							
Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	. ^	
EP4CE22E22C9L	1.0V	22320	80	608256	132		
EP4CE22E22I7	1.2V	22320	80	608256	132		
EP4CE22E22I8L	1.0V	22320	80	608256	132		
EP4CE22F17A7	1.2V	22320	154	608256	132		
EP4CE22F17C6	1.2V	22320	154	608256	132		
EP4CE22F17C7	1.2V	22320	154	608256	132		
EP4CE22F17C8	1.2V	22320	154	608256	132		
EP4CE22F17C8L	1.0V	22320	154	608256	132		
EP4CE22F17C9L	1.0V	22320	154	608256	132		
EP4CE22F17I7	1.2V	22320	154	608256	132		
EP4CE22F17I8L	1.0V	22320	154	608256	132	~	
<	1.1.216	77570	154	000700		>	

Figure 4-6 Device and Options

Choose unused pins. Reserve all unused pins: Choose the As input tri-stated option. See Figure 4-7.



🐇 Device and Pin Options - my	_first_fpga 🛛 🗙					
Category:						
General	Unused Pins					
Configuration Programming Files Unused Pins Dual-Purpose Pins Capacitive Loading Board Trace Model I/O Timing Voltage Pin Placement Error Detection CRC CVPCIe Settings	Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.  Reserve all <u>u</u> nused pins: As input tri-stated					
	Reset					
	OK Cancel Help					

Figure 4-7 Setting unused pins

Click twice OK.

4. In the Processing menu, choose Start Compilation. After the compile, Choose 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 unlighted.





### 5.1 Headquarter & Branches

• Tel +886-3-5750880

ter asiC

- Fax +886-3-5726690
- Add 9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan
- Email sales@terasic.com / support@terasic.com