

My First FPGA Design Tutorial



101 Innovation Drive San Jose, CA 95134 (408) 544-7000 http://www.altera.com

TU-01002-1.3

Document Date:

July 2008

Copyright © 2008 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the ap-

changes to any products and services at any time without nonce. After assumes to responsibility or hability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.







Contents

How to Contact Altera	
Typographic Conventions	v
Introduction	1–1
Design Flow	1–1
Before You Begin	1–2
What You Will Learn	1–2
Get Started	1–3
Assign the Device	1–6
Design Entry	
Add a PLL Megafunction	. 1–14
Add a Multiplexer	. 1–28
Assign the Pins	. 1–35
Create a Default TimeQuest SDC File	. 1–38
Compile Your Project	39
Program the Device	
Verify in Hardware	
Next Steps	



About this Tutorial

This tutorial provides comprehensive information that will help you understand how to create an Altera[®] FPGA design and run it on your development board.

How to Contact Altera

For the most up-to-date information about Altera products, refer to the following table.

Information Type	Contact (1)
Technical support	www.altera.com/mysupport/
Technical training	www.altera.com/training/ custrain@altera.com
Product literature	www.altera.com/literature/
Altera literature services	literature@altera.com
FTP site	ftp.altera.com

Note to table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: Save As dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: f _{MAX} , \qdesigns directory, d: drive, chiptrip.gdf file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design.</i>
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA} , $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type.
	Example: <i><file name=""></file></i> , <i><project name="">.pof</project></i> file.

Visual Cue	Meaning
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
Courier type	Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix n, e.g., resetn.
	Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\qdesigns\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
••	Bullets are used in a list of items when the sequence of the items is not important.
\checkmark	The checkmark indicates a procedure that consists of one step only.
I	The hand points to information that requires special attention.
CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
WARNING	The warning indicates information that should be read prior to starting or continuing the procedure or processes
4	The angled arrow indicates you should press the Enter key.
•••	The feet direct you to more information on a particular topic.



1. My First FPGA Design

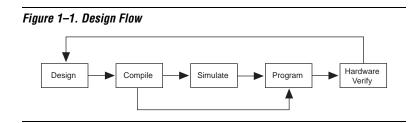
July 2008

Introduction

Welcome to Altera and the world of programmable logic! This tutorial will teach you how to create a simple FPGA design and run it on your development board. The tutorial takes less than an hour to complete. The following sections provide a quick overview of the design flow, explain what you need to get started, and describe what you will learn.

Design Flow

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 create the digital circuit that is implemented inside the FPGA. The flow then proceeds through compilation, simulation, programming, and verification in the FPGA hardware (see Figure 1–1).



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, RTL and timing. RTL (or 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.

- P
- See "Next Steps" on page 1–45 for links to further information about simulation.

Before You Begin

This tutorial assumes the following prerequisites:

- You generally know what an 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 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 an Altera Cyclone[®] III, Stratix[®] III, or Arria[™] GX Development Board (or equivalent) 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.
 - Installed the USB-Blaster[™] driver, which allows you to program the FPGA on the development board with your own design.

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 button—This design is easy to create and gives you visual feedback that the design works. Of course, you can use your development 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.

Become familiar with Quartus II design tools—This tutorial will not
make you an expert, but at the end, you will understand basic
concepts about Quartus II projects, such as entering a design using a
schematic editor and HDL, compiling your design, and
downloading it into the FPGA on your development board.

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 FPGA development board.

For information about Altera training classes (both on-line and in person), go to the Altera web site at mysupport.altera.com/etraining/ or contact your local Altera sales representative.

Get Started 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.

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

The New Pi following:	roject Wizard helps you create a new project and preliminary project settings, including the
	Project name and directory Name of the top-level design entity Project files and libraries Target device family and device EDA tool settings
the Settings	ange the settings for an existing project and specify additional project-wide settings with command (Assignments menu). You can use the various pages of the Settings dialog box tionality to the project.
🗆 Don't sk	now me this introduction again

Figure 1–2. 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, for example, c:\altera\my_first_fpga.
 - File names, project names, and directories in the Quartus II software cannot contain spaces.
 - 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_top. See Figure 1–3.

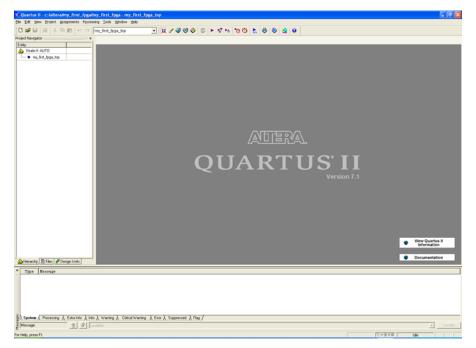
	oject?	
c:\altera\my_first_fpga		-
What is the name of this project?		
my_first_fpga		
What is the name of the top-level desi exactly match the entity name in the d	n entity for this project? This name is case sign file.	e sensitive and mu
my_first_fpga_top		

Figure 1–3. Project Information

- d. Click Finish.
- The wizard has several other pages after this one; however, for this tutorial you do not need to make changes to these pages. For more information on the options available in these pages, refer to the **Quartus II Handbook**.
- 4. When prompted, choose **Yes** to create the **my_first_fpga** project directory.

Congratulations! You just created your first Quartus II FPGA project. See Figure 1–4.

Figure 1-4. my_first_fpga Project



Assign the Device

In this section, you will assign a specific FPGA device to the design and make pin assignments. To assign the device, perform the following steps.

- 1. Choose **Assignments > Device**.
- 2. Under **Family**, choose the device family that corresponds to the device on the development board you are using.
- 3. Under Available devices, choose the device given in Table 1–1.

Table 1–1. Available Device Settings						
Development Board	Device Family	Package	Pin Count	Setting		
Arria GX Development Board	Arria GX	FBGA	780	EP1AGX60DF780C6		
Stratix III Development Board	Stratix III	FBGA	1,152	EP3SL150F1152C3		
Cyclone III Starter Board	Cyclone III	FBGA	324	EP3C25F324C8		
Cyclone III Development Board	Cyclone III	FBGA	780	EP3C120F780C7		

See Figure 1–5.

Figure 1	1-5. 8	Specify	the	Device	Example
----------	--------	---------	-----	--------	---------

General	Device						
 Files Libraires Device Operating Settings and Conditions Complainon Process Settings EDA Tool Settings Flandysis & Synthesis Settings Filter Settings Filter Settings Timing Analysis Settings Assembler Design Assistant 	Control Settings Entity: [Ucyclone III Device & Pin Options Synthesis Settings Target device Calud device selected by the Fitter Control Setting Control Setting Control Setting					ailable dev Any Any de: Any dvanced o	v
- SignalTap II Logic Analyzer - Logic Analyzer Interface							
Simulator Settings	Available devices:	10	Lur.	1	1.11		1.00
PowerPlay Power Analyzer Settings	Name EP3C25F324C6 EP3C25F324C7 EP3C25F324C8	Core v 1.2V 1.2V 1.2V	LEs 24624 24624 24624	User I/ 216 216 216	Memor 608256 608256 608256	Embed 132 132 132	4 4 4
	 Migration compatibility 		un Companion	device			>
	Migration Devices.	-	HardCopy		o HardCopy I	l device re	• sources

4. Click OK.

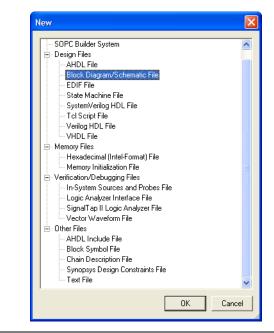
Design Entry

In the design entry phase, you use RTL or schematic entry to create the logic to be implemented in the device. You also make pin assignments, including pin placement information, and timing constraints that might be necessary for building a functioning design.

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.

- Choose File > New > Block Diagram/Schematic File (see Figure 1–6) to create a new file, Block1.bdf, which you will save as the top-level design.
- 2. Click OK.

Figure 1–6. New BDF



- 3. Choose **File > Save As** and enter the following information (see Figure 1–7).
 - **File name:** my_first_fpga_top
 - Save as type: Block Diagram/Schematic File (*.bdf)

Figure 1–7. Saving the BDF

Save jn	my_first_fpg	ja		-
My Recent Documents	ia db			
My Computer My Network Places	File <u>n</u> ame:	my first (pga.top	<u> </u>	Save
	Save as type:	Block Diagram/Schematic		Cancel

4. Click **Save**. The new design file appears in the Block Editor (see Figure 1–8).

Figure 1–8. Blank BDF

🗃 🖬 🍯 X 🖻 🖻 🕫 o	Processing Tools Window Help	my_first_[pga_top.bdf]			- 6
ct Nevigetor		1680 0 + 5 m -	00 🕻 🗠 🗞		
by				 	
Stratix II: AUTO					
my_first_lpgs_top					
	- Þ				
	Α				
	Ð				
	7				
	7				
	1				
	<u>+</u>				
	1				
	0,				
	45				
	4				
	£				
	0				
	Y				
	N				
	`				
er meriken 🚺 Eduar an 🖉 Danairen Harita					
erarchy 🗈 Files 🖋 Design Units					

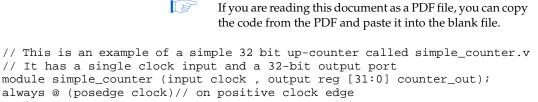
- Add HDL code to the blank block diagram by choosing File > New > Verilog HDL File.
- 6. Click **OK** to create a new file **Verilog1.v**, which you will save as **simple_counter.v**.
- Select File > Save As and enter the following information (see Figure 1–9).
 - File name: simple_counter.v
 - Save as type: Verilog HDL File (*.v, *.vlg, *.verilog)

Figure 1–9. Saving the Verilog HDL File

Save As					X
Save in My Recent Documents Desktop My Documents	irst_fpg	a	•		7
My Network Places	File <u>n</u> ame: Save as <u>t</u> ype:	simple_counter.v Verilog HDL File ✓ Add file to cu	(*.v;*.vlg;*.verilog)	•	<u>S</u> ave Cancel

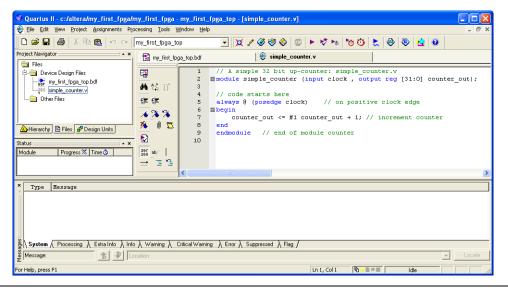
The resulting empty file is ready for you to enter the Verilog HDL code.

8. Type the following Verilog HDL code into the blank **simple_counter.v** file (see Figure 1–10).



begin counter_out <= #1 counter_out + 1;// increment counter end endmodule// end of module counter

Figure 1–10. simple_counter.v



- 9. Save the file by choosing **File > Save**, pressing Ctrl + s, or by clicking the floppy disk icon.
- 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 1–11).

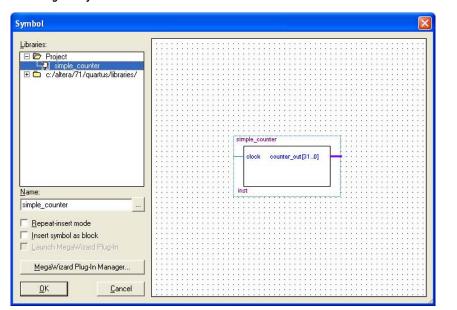
Figure 1–11. Symbol File Created

Quartus	II 🛛 🔀
(į)	Create Symbol File was successful
	OK]

- 11. Click OK.
- 12. To add the **simple_counter.v** symbol to the top-level design, click the **my_first_fpga_top.bdf** tab.

- 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

Figure 1–12. 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 1–13.

Figure 1–13. Placing the Symbol

🖁 Quartus II - c:/altera/my_first_fpga/my_first_fpga - my_first_fpga_top - [my_first_fpga_top.bdf*]					
🔁 Eile Edit View Project Assignments Pro	cessing <u>T</u> ools <u>W</u> indow <u>H</u> elp		_ & ×		
🛛 🗅 🖨 🔚 🛛 🎒 👗 🐚 💼 🗠 🗠 🗍	my_first_fpga_top 📃 🔀	(200) *** 💿 🖉 🖉	≿ 🐵 🕸 👱 🞯		
Project Navigator * *	📸 my_first_fpga_top.bdf*	simple_counter.v	🕘 Compilation Report		
Files F			Simple source U Global courter_ox[D1.0] mat		
Type Message Type Message Info: Type Info: Type Info					
For Help, press F1		165, 158 🔀 🗝	idle ///		

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

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 (which is different for different development boards) 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.

 Choose Edit > Insert Symbol or click Add Symbol on the toolbar (
 D).
 2. Click **Megawizard Plug-in Manager**. The MegaWizard[®] Plug-In Manager appears (see Figure 1–14).

Figure 1–14. MegaWizard Plug-In Manager

		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>Create a new custom megafunction variation</u> © Edit an existing custom megafunction variation © Copy an existing custom megafunction variation Copyright © 1991-2007 Altera Corporation
--	--	---

- 3. Click Next.
- 4. In **MegaWizard Plug-In Manager [page 2a]**, specify the following selections (see Figure 1–15):
 - a. Choose I/O > ALTPLL.
 - b. Under Which device family will you be using?, choose the device family the corresponds to the device on your development kit 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.

Vhich megafunction would you like to customize? Select a megafunction from the list below	Which device family will you be Cyclone III using?
Arithmetic Arithmetic Ommunications	Which type of output file do you want to create?
	C AHDL C VHDL
ALT2GX8	Verilog <u>H</u> DL What name do you want for the gutput file? Browse d.\altera\my_first_fpga\pll
ALTCLKLOCK ALTDDIO_BIDIR ALTDDIO_OUT ALTDQ ALTDQS ALTDQS ALTDQS ALTGS8 ALTIOBUF ALTGK8 ALTIOBUF ALTMEMPHY ALTOCT ALTCL ALTC	Return to this page for another create operation Note: To compile a project successfully in the Quartus II software your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a use library specified in the User Libraries page of the Settings dialog box (Assignments menu). Your current user library directories are:
ALTPLL_RECONFIG ALTREMOTE_UPDATE MAX II oscillator	

Figure 1–15. 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 1–16).
 - a. Confirm that the **Currently selected device family** option shows the device family that corresponds to the development board you are using.
 - b. Under Which device speed grade will you be using?, type the value given in Table 1–2.

Table 1–2. Speed Grade Settings				
Development Board	Setting			
Arria GX Development Board	6			
Stratix III Development Board	3			
Cyclone III Starter Board	8			
Cyclone III Development Board	7			

c. Under **What is the frequency of the inclock0 input?**, type the value given in Table 1–3.

Table 1–3. Input Clock Frequency Settings			
Development Board	Setting (MHz)		
Arria GX Development Board	100		
Stratix III Development Board	50		
Cyclone III Starter Board	50		
Cyclone III Development Board	50		

- d. Ensure that the units are **MHz** (default).
- e. Click Next.

legawizar	d Plug-in Manage	r [page 3	or 14]					
2	ALTPLL							
Name of	_	,					About	Documentation
Parameter Settings	2 PLL Reconfiguration							
Seneral/Modes	_	andwidth/SS	L Clock swit	rchover				
		anannaan,oo	Clock Shie					
	pll				Currently selected	device family:	Cyclone :	
	bu						🗹 Match	project/default
jnclk0			:0, <u>аы</u>	le to implement the	requested PLL			
I avaaat II	nclk0 frequency: 50.000 MH Operation Mode: Normal	2 locke		ie to implement die	Frequesced FLL			
l I	Clk Ratio Ph (dg) DC (%)							
	c0 1/1 0.00 50.00		6	ieneral				
L		Cyclone III						
L		Cyclone III		Which device spee	d grade will you be using?		7	~
				Use military ten	nperature range devices only			
				What is the freque	ncy of the inclock0 input?		50	MHz 🗸
				Set up PLL in L	/DS mode	Data rate:	Not Avai	lable 🗸 Mbps
			_ P	LL type				
				Which PLL type w	ill uou be using?			
				O Fast PLL	n you bo using.			
				O Enhanced PLL				
				Select the PLL t	:ype automatically			
			-0	peration mode				
					utputs be generated?			
					ick path inside the PLL			
				💿 In Normal M	ode			
				 In Source-S 	ynchronous Compens <u>a</u> tion M	ode		
				○ In <u>Z</u> ero Dela	ay Buffer Mode			
				Connec	t the fbmimic port (bidirection	nal)		
				○ <u>W</u> ith no con				
				O Create an 'fbin'	input for an external feedba	ack (<u>E</u> xternal F	eedback Mode	e)
				Which output clock	will be compensated for?		c0 🗸	
						Cancel	< <u>B</u> ack <u>N</u> e	ext > Einish

Figure 1–16. 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 1–17 for an example.

MegaWizard Plug-In Manager [page 4	of 14]
altpll	About Documentation
Parameter PLL 3 Output Settings Reconfiguration Clocks	4 EDA 5 Summary
General/Modes Inputs/Lock Bandwidth/SS	Clock switchover
pll Incik0 frequency: 50.000 MHz Operation Mode: Kornal Cik Ratio Ph (dg) DC (13) co 1/1 0.00 60.00 Cyclone III	Able to implement the requested PLL Cptional inputs Create an 'plena' input to selectively enable the PLL Create an 'areget' input to asynchronously reset the PLL Create an 'pridena' input to selectively enable the phase/freq. detector Lock output Create 'locked' output Enable self-reset on loss of lock Hold 'locked' output to 'prior the PLL initializes Advanced PLL parameters Using these parameters is recommended for advanced users only Create output file(s) using the 'Advanced' PLL parameters Configurations with output clock(s) that use cascade counters are not supported

Figure 1–17. MegaWizard Plug-In Manager [page 4 of 14] Selections

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

a. Under **Clock division factor**, use the up/down arrows or enter the value given in Table 1–4.

Table 1–4. Clock Division Settings				
Development Board	Setting			
Arria GX Development Board	20			
Stratix III Development Board	10			
Cyclone III Starter Board	10			
Cyclone III Development Board	10			

See Figure 1–18 for an example that uses 10 as the division factor.

MegaWizard Plug-In Manager [page	8 of 14]	
ALTPLL		About Documentation
1 Parameter 2 PLL 3 Output Settings Reconfiguration Clocks	t 4 EDA 5 Summary	
clk c0 clk c1 clk c2 clk c3 clk c4		
	c0 - Core/External Output Clock	
pli	Able to implement the requested PLL	
jnclk0 c	J. 🗹 Use this clock	
inclik0 frequency: 50.000 MHz Operation Mode: Normal	→Clock Tap Settings	
Clk Ratio Ph (dg) DC (%)		Requested settings Actual settings
c0 1/10 0.00 50.00	 Enter output clock frequency: 	100.0000000 MHz V 5.000000
Cyclone III	 Enter output clock parameters: 	
	Clock multiplication factor	
	Clock division factor	10 << Copy 10
	Clock phase shift	0.00 🔶 deg 🗸 0.00
	Phase shift step resolution(ps)	
	Clock duty cycle (%)	50.00 \$
	More Details >>	
		Per Clock Feasibility Indicators
		cO c1 c2 c3 c4
		Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

Figure 1–18. MegaWizard Plug-In Manager [page 7 of 14] Selections

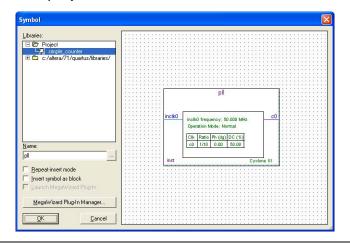
- 9. Click Finish.
- 10. The wizard displays a summary of the files it creates (see Figure 1–19). Click **Finish** again.

Figure 1–19. Wizard-Created Files

MegaWizard Plug-In Manager [page 14	of 14] Summary	
3		About Documentation
Parameter PLL 3 Output	4 EDA 5 Summary	
Settings Reconfiguration Clocks		
pil inclk0 frequency: 50.000 MHzC0	automatically generated, a	to generate. A gray checkmark indicates a file that is ind a red checkmark indicates an optional file. Click cted files. The state of each checkbox is maintained in lug-In Manager sessions.
Operation Mode: Normal Clik Ratio Ph (4g) DC (%) c0 1/10 0.00 50.00	directory: d:\altera\my_first_fpga\	anager creates the selected files in the following
Cyclone III	File	Description
	i pl.v i pl.ppf	Variation file PinPlanner ports PPF file
	D pll.inc	AHDL Include file
	D pll.cmp	VHDL component declaration file
	i pll.bsf □ pll_inst.v	Quartus II symbol file Instantiation template file
	i pi_nscv i pi bb.v	Verilog HDL black-box file
	☑ pl_waveforms.html	Sample waveforms in summary
	L. pll_wave*.jpg	Sample waveform file(s)
		Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

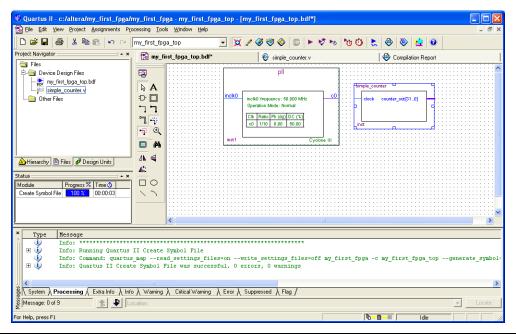
The Symbol window opens, showing the newly created PLL megafunction. See Figure 1–20.

Figure 1–20. pll Symbol



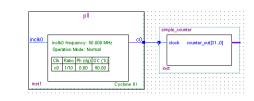
11. Click **OK** and place the pll symbol onto the BDF to the left of the simple_counter symbol. You can move the symbols around by holding down the left mouse button, helping you ensure that they line up properly. See Figure 1–21.

Figure 1–21. Place the pll Symbol



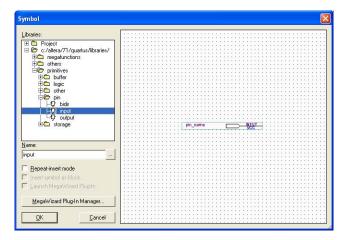
- 12. Move the mouse so that the cursor (also called the selection tool) is over the pll symbol's c0 output pin. The orthogonal node tool (cross-hair) icon appears.
- 13. Click and drag a bus line from the c0 output to the simple_counter clock input. This action ties the pll output to the simple_counter input (see Figure 1–22).





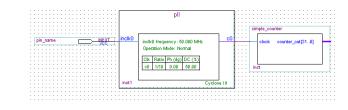
- 14. Add an input pin and an output bus with the following steps:
 - a. Choose Edit > Insert Symbol.
 - Under Libraries, select quartus/libraries > primitives > pin > input. See Figure 1–23.
 - c. Click OK.
 - If you need more room to place symbols, you can use the vertical and horizontal scroll bars at the edges of the BDF window to view more drawing space.

Figure 1–23. 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 1–24.

Figure 1–24. Connecting Input Pin



f. Change the pin name by double-clicking pin_name and typing osc_clk (see Figure 1–25). This name correlates to the oscillator clock that is connected to the FPGA.

Figure 1–25. Change Input Pin Name

Pin Properties General Format]	
To create multi	, ole pins, enter a name in AHDL bus notation (for example, r enter a comma-separated list of names.	
<u>P</u> in name(s):	osc_clk	
<u>D</u> efault value:	VCC	
	OK	el

- g. Using the Orthogonal Bus tool (1), 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.
- h. Right-click the new output bus line and choose **Properties**.

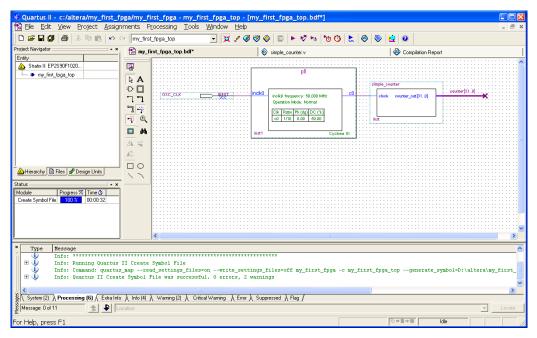
i. Type counter[31..0] as the bus name (see Figure 1–26). The notation [X..Y] is the Quartus II method for specifying the bus width in BDF schematics, where X is the most significant bit (MSB) and Y is the least significant bit (LSB).

Figure 1–26. Change Output Bus Name

Bus Properties				X
General Font	Format			
<u>N</u> ame:	counter[310]			
□ <u>H</u> ide name	in block design file			
			OK	Cancel

j. Click OK. Figure 1–27 shows the BDF.

Figure 1–27. BDF



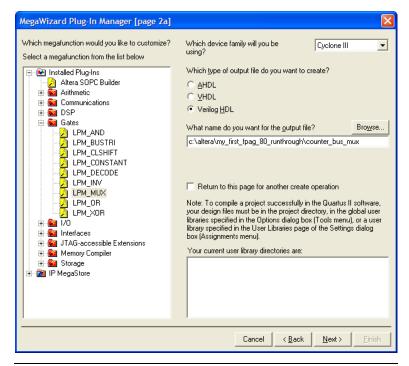
Add a Multiplexer

This design uses a multiplexer to route the simple_counter output to the LED pins on the 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 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** (see Figure 1–28).

6. Click Next.

Figure 1–28. 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 result output be?, select 4 (see Figure 1–29).

MegaWizard Plug-In Manager - LPM_MUX [page 3 of 5]						
2	LPM_MUX Version 7.1	About				
1 Parameter 2 Settings	EDA 3 Summary					
counter_bu	s_mux c	urrently selected device family: Cyclone III				
data1×[30]	el Hov	v many 'data' inputs do you want? 2 💌 v wide should the 'data' input and result' output buses be? bits				
	Do	ou want to pipeline the multiplexer?				
	[No Yes, I want an output latency of Clock cycles Create an asynchronous Clear input Create a Clock Enable input				
Resource Usag	e	Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish				

Figure 1–29. Ipm_mux Settings

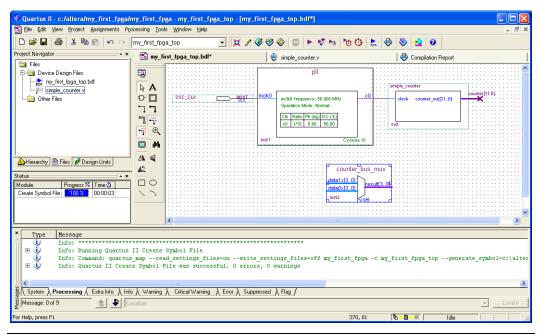
- 9. Click Next.
- 10. Click **Finish** twice. The Symbol window appears (see Figure 1–30 for an example).

Figure 1–30. lpm_mux Symbol

Libraries:	
E D Pojet D c/alexe/71/oustuu/libraie D c/	ev counter_bus_mux detext3.00 result3.00 result3.00
Name:	inst Sel
counter_bus_mux	
<u>Repeat insert mode</u> Insert symbol as block Launch MegaWizard Plug-in	
MegaWizard Plug-In Manager	

- 11. Click **OK**.
- 12. Place the counter_bus_mux symbol below the existing symbols on the BDF. See Figure 1–31.

Figure 1–31. Place the lpm_mux Symbol



- 13. Add input buses and output pins to the counter_bus_mux symbol as follows:
 - Using the Orthogonal Bus tool, draw bus lines from the data1x[3..0] and data0x[3..0] input ports to about 6 to 8 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 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 1–32 shows the renamed buses.

Figure 1–32. Renamed counter_bus_mux Bus Lines



P

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

- 14. Choose Edit > Insert Symbol.
- 15. Under Libraries, double-click quartus/libraries/ > primitives > pin> output (see Figure 1–33).

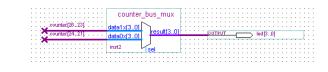
Figure 1–33. Choose an Output Pin

Libraries:	
🖽 🗀 Project	• • • • • • • • • • • • • • • • • • • •
E C c:/altera/71/quartus/libraries/	
🕀 🗀 megafunctions	
🗄 🖨 others	I
Ė 🗁 primitives	
E buffer	
🕸 🗖 logic	
🗄 🖨 other	
🛱 🗁 pin	
i i-t⊅ bidir	
Cpr input	
U U output	
± € storage	DUTPUT pin name
in the storage	por laure
Name:	
output	
Eppeat-insert mode	
Insert symbol as block	
Launch MegaWizard Plug-In	
1	
MegaWizard Plug-In Manager	
OK Cancel	



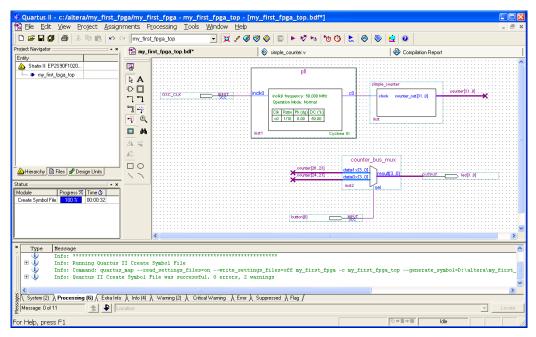
- 17. Place this output pin so that it connects to the counter_bus_mux result[3..0] bus output line.
- 18. Rename the output pin as led[3..0] as described in steps 13 c and d. (see Figure 1–34).

Figure 1–34. Rename the Output Pin



- 19. Attach an input pin to the multiplexer select line using an input pin:
 - a. Choose Edit > Insert Symbol.
 - Under Libraries, double-click quartus/libraries/ > primitives > pin > input.
 - c. Click OK.
- 20. Place this input pin below counter_bus_mux.
- 21. Connect the input pin to the counter_bus_mux sel pin.
- 22. Rename the input pin as button[0] (see Figure 1–35).

Figure 1–35. Adding the button[0] Input Pin



You have finished adding symbols to your design.

You can add notes or information to the project as text using the the Text tool on the toolbar (indicated with the A symbol). For example, you can add the label "OFF = SLOW, ON = FAST" to the button[0] input pin and add a project description, such as "My First FPGA Project."

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 button[0] and osc_clk 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 1–36.

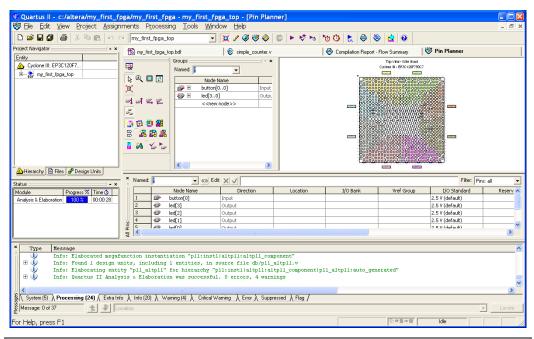


Figure 1–36. Pin Planner Example

2. In the **Location** column next to each of the six node names, add the coordinates (pin numbers) as shown in Table 1–5 for the actual values to use with your board.

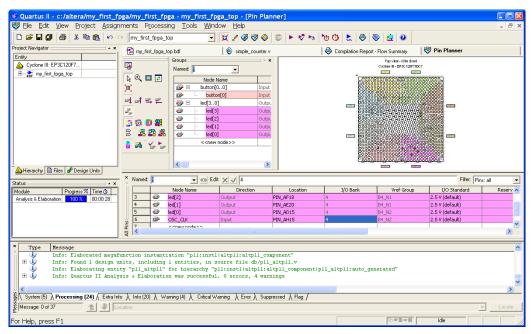
Table 1–5. Pin Information Settings							
Pin	Arria GX Development Board	Stratix III Development Board	Cyclone III Starter	Cyclone III Development Board			
button[0]	A19	B17	F1	AD7			
led[3]	B13	A23	N9	AD19			
led[2]	C13	B23	N12	AF18			
led[1]	A11	C23	P12	AE20			
led[0]	A12	F21	P13	AD15			
osc_clk	U28	Т33	V9	AH15			

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 1–37.

F

To provide more detailed information about the development boards, Altera provides board reference manuals and schematics for all development boards. These documents provide complete information about the pinouts for the devices on the boards.





You are finished creating your Quartus II design!

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:

- 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 osc_clk osc_clk
derive_pll_clocks
derive_clock_uncertainty

4. Save this file as **my_first_fpga_top.sdc** (see Figure 1–38).

Figure 1–38. Default SDC

ile <u>E</u> dit <u>V</u> iew <u>P</u> r	oject <u>T</u> ools	Window		
a M 45 0 F F 4 % % % 1 5 2 8 ∞ →	1 2 3 4 5	create_clock -period 20.000 -name osc_clk osc_clk derive_pll_clocks derive_clock_uncertainty		
r Help, press F1	•		Col 1	•

P

Naming the SDC with the same name as the top-level file except for the .sdc extention 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.



For more advanced information about timing settings and the TimeQuest timing analyzer, refer to the Quartus II Verification and Simulation page on Altera's website at www.altera.com/products/software/products/quartus2/verification/ qts-verification.html.

Compile Your Project

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.

If you want to store SOFs in memory devices (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 1–39).

Figure 1–39. Compilation Message Example

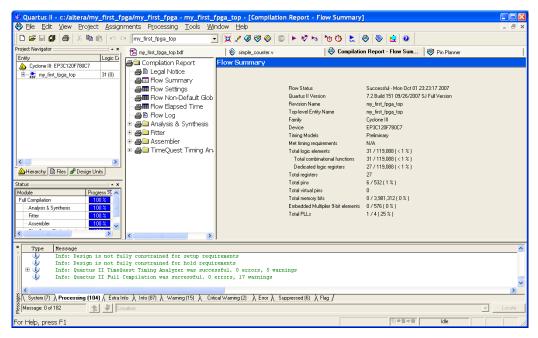
Quartus II - c:/altera/my_first_fpga/my_first_fpga - my_first_f Ele Edit View Project Assignments Processing Tools Win		n Report - Flow Summary]	- = ×
🖸 🖆 🖶 🕼 🐇 🖏 🛍 🗠 🕫 🕅 my_first_fpga_top 📃	💢 🖌 🏈 🧇 🖉 🚳	• • • • • • • • • •	🗢 🕸 🖻 🙆	
Project Navigator	🛛 🕸 simple_counter.v	🛛 🕀 Compilatio	n Report - Flow Sum 😻 Pin Planner	1
A Cychone III: EP3(120F7 B - gr mg_first_lbga_top B - gr mg_first_gr mg_first_lbga_top	Flow Summary	Flow Status Quartus II Version Revision Name Toplevel Entity Name Family Device Timing Models Met timing requirements Total logic elements Total combinational functions Dedicated logic registers Total ergisters Total ergisters Total ergisters Total virtual pins Total virtual pins Total virtual pins Total virtual pins Total virtual pins Total PLLs	In progress - Mon Oct 01 23:21:36 2007 7.2 Build 151 09/26/2007 SJ Full Version mv_frit_pop_top Cyclore III EP3C1207780C7 Prefininary N/A 31 31 31 27 27 6 6 0 0 1	
Type Message Info: Stating register packing: Info: Finished register packing: Info: Fitter placement preparation operations ending: Info: Fitter placement preparation operations ending: Info: System (5) Processing (50) Fitalno Info: Fitcondore	ng		<u>_</u>	Locate
For Help, press F1			◎ ⇒ ■ ⇒ ■ 32% ₩ 00:01:21	

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 1–40.

Figure 1–40. Compilation Report Example



Program the 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:

- 1. Connect the power supply cable to your board and to a power outlet.
- 2. For Cyclone III and Stratix III development boards, connect the USB cable to the board. For the Arria GX board, connect the USB-Blaster (included in your development kit) to J4 and the USB cable to the USB-Blaster. Connect the other end of the USB cable to the host computer.
 - P

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

3. Turn the board on using the on/off switch (SW1).

For the Arria GX development board, before opening the device programmer, place the MAXII and HSMC slide switches in the bypass position. The default position (without an HSMC card installed) is HSMC in bypass mode and MAXII in chained mode.

Program the FPGA using the following steps.

1. Choose **Tools > Programmer**. The Programmer window opens. See Figure 1–41.

Figure 1–41. Programmer Window

	my_first_fpga_t					≿ 🗶 🧶 👱 🙆		
oject Navigator X × X	🔝 my_first_fpga_t		iple_counter.v	🛛 🗢 Compila	ation Report - Flo	w Su 🛛 😻 Pin Planner	📔 Chain1.cdf	
Cyclone III: EP3C120F780C7	📩 Hardware Setup	No Hardware				Mode: JTAG	Progress: 0	%
🖅 💦 my_first_fpga_top 31 (0)	Enable real-time	SP to allow background p	rogramming (for MAX II	devices)				
	M a Start	File	Device	Checksum	Usercode	Program/ Verify Blank- Configure Verify Check	Examine Security Erase	ISP CLAMP
	🖬 Stop		•					
	Auto Detect							
	🗙 Delete							
	🍰 Add File							
	Change File							
Hierarchy 🖹 Files 🗗 Design Units	Save File							
us	Add Device							
dule Progress % 🔨	Alle Up							
Il Compilation 100 %	JB Down							
Fitter								
Assembler 100 %								
Type Message Info: Design is not fu	lly constrained	for setup require	ients					
 Info: Design is not fu Info: Quartus II TimeQ 	lly constrained	for hold requirem	ents					
info: Quartus II fimeQ								
<								>
	nto à Info(97) à u	aming (15) 👌 Critical V	Varning (2) X Error	Suppressed (6)	λ Elag /			

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 1–42.

Figure 1–42. Hardware Settings

Currently selected hardware:	USB-Blaster [U	SB-01	•
Available hardware items:	, .		
Hardware USB-Blaster	Server Local	Port USB-0	Add Hardware

- 4. Click Close.
- 5. If the file name in the **Programmer** does not show **my_first_fpga_top.sof**, click **Add File**.
- 6. Select the **my_first_fpga_top.sof** file from the project directory (see Figure 1–43).

Figure 1–43. Add Programming File

elect Programmir	ng File					×
Look <u>i</u> n:	🗀 my_first_fpga			• +	€ 🔺 🔳	
My Recent Documents Desktop My Documents My Computer My Network	db rwyfirst_ppa_top.pof my_first_ppa_top.cof					
Places	File <u>n</u> ame:	my_first_fpga_t	op.sof		•	<u>O</u> pen
	Files of type:	Programming F	Files (*.sot*.pot*	.jam;*.jbc;*.e	kp;*.jic ▼	Cancel

- 7. Click Open.
- Turn on the Program Configure option that corresponds to the my_first_fpga_top.sof file.
- 9. Click Start. The file downloads to the development board.

The progress bar shows the download status; the status is 100% when downloading completes. See Figure 1–44.

Quartus II - C:/altera	/kits/CIII_quick_bri	ng_up/my_fi	rst_fpga/my_i	iirst_fpga <mark> - my</mark> _	first_fpga_to	op - [my_fin	st_fpga_top.	cdf*]		_ 🗆 X
Eile Edit View Pro	ject <u>A</u> ssignments P <u>r</u>	ocessing <u>T</u> oo	s <u>W</u> indow <u>H</u>	elp						_ & ×
D 📽 🖬 @ X 🖻 6	∎ ⊷ ⇔ my_first_f	oga_top	- 🐹 🗸 🤇	ا 🗇 🍪 🕉 🖉	• 🕏 🏍 🏷	🕐 🚼 🤄) 🕹 🕹	0		
Project Navigator * *	🖹 my_first_fpga_top	bdf*	🍣 Compilati	on Report - Flow S	umm 😻 Pir	n Planner		📔 my_firs	t_fpga_top.c	df*
A Cyclone II: EP2C35F6	🔔 Hardware Setup	USB-Blast	er [USB-0]							
⊞ 💦 my_first_fpga_top	Mode:	JTAG								•
	Progress:					100 %				
	🔲 Enable real-time IS	P to allow back	ground program	ming (for MAX II de	vices)					
	M Start	File		Device	Checksum	Usercode	Progra Ve Configu	erify Blank- Check Exe	ami Secur Bit	Erase ISF CLAN
Status ***	🖷 Stop	my_first_f	oga_top.sof	EP2C35F672	002E664D	FFFFFFF				
Module Full Compilation	H Auto Detect									
- Analysis & Synthe	X Delete									
Partition Merge										
Assembler	Add File									
	👺 Change File	•								
× ③ Info: Start	ed Programmer op	eration at	Wed Jan 17	18:03:13 20	07					
	guring device in									
4	e 1 contains JTA									
Info: Configuration succeeded 1 device(s) configured										
 Info: Successfully performed operation(s) Info: Ended Programmer operation at Wed Jan 17 19:03:15 2007 										
🐶 Into: Ended	Programmer oper	ation at w	ed Jan I/ I	8:03:15 2007						
System / Processing) Extra Info) Info) V	√arning	al Warning ∫ E	Error 👌 Suppress	ed /					
Message: 0 of 6		ocation:							~	Locate
For Help, press F1							№-D+0	Idle		

Figure 1–44. 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.

Verify in 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:						
	 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 [2623]. 						
	The LEDs are active low, therefore, when counting begins all LEDs are turned on (the 0000 state).						
	2. Press and hold Button 1 on the development board and observe that the LEDs advance more quickly. Pressing this button causes the design to multiplex using the faster advancing part of the counter (bits [2421]).						
Next Steps	Altera provides many tutorials and reference material that you can use to further your knowledge of FPGA design. The information on the following web pages will help you learn more about Altera tools and products.						
	 www.altera.com/education/univ/unv-index.html www.altera.com/education/univ/materials/manual/unv-lab- manual.html 						
	 www.altera.com/literature/lit-qts.jsp www.altera.com/end-markets/refdesigns/ref-index.jsp www.altera.com/support/examples/exm-index.html www.altera.com/corporate/contact/con-index.html mysupport.altera.com/etraining/ 						
	The Altera etraining page provides tutorials for each of the steps covered in this document. Additionally, it provides simulation tutorials, such as:						
	Using the Quartus II Software: SimulationUsing the Quartus II Software: Timing Analysis						
	 Constraining and Analyzing Timing for Source Synchronous Circuits with TimeQuest Validating Performance with the TimeQuest Static Timing Analyzer 						