My First FPGA Design Tutorial

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

TU-01002-1.0

Contents

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: *AN 75: High-Speed Board Design.*
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: *<file name>*, *<project name>***.pof** 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.
✓	The checkmark indicates a procedure that consists of one step only.
☞	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
↵	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

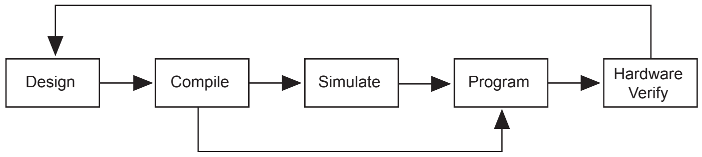
Introduction

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, and programming and verification in the FPGA hardware (see Figure 1–1).

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, 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.

See "Next Steps" on page 1–43 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 Starter 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 Cyclone III 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 4 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 the Cyclone III Starter Board.

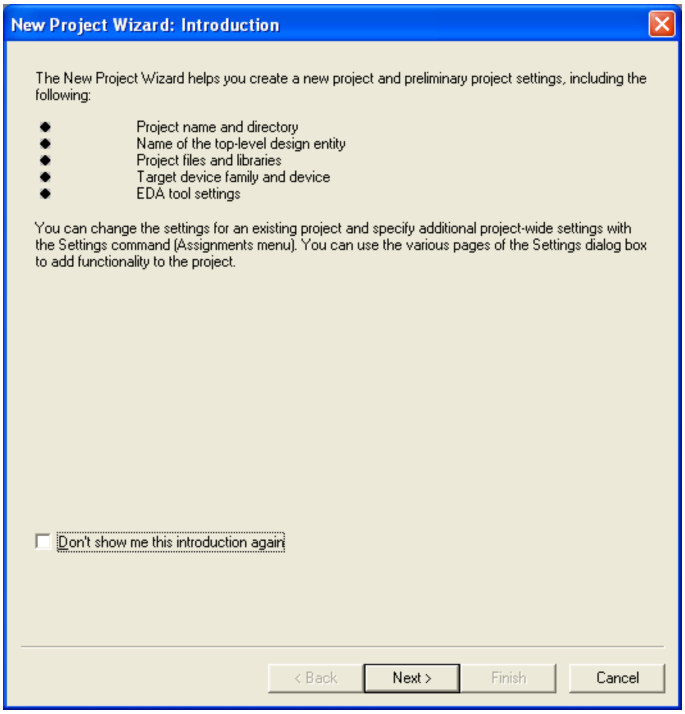
For information about Altera training classes (both on-line and in person), go to the Altera web site at mysupport.altera.com/etraining/ or contact your local Altera sales representative.

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.

Figure 1–2. New Project Wizard: Introduction



New Project Wizard: Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

☐ Don't show me this introduction again

2. Click **Next.**

3. Enter the following information about your project:

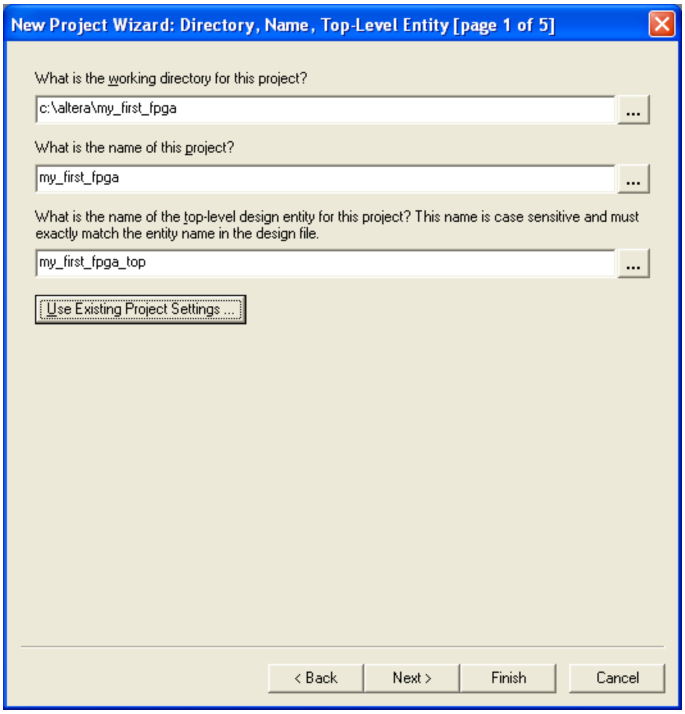
   a. **What is the working directory for this project?** Enter a directory in which you will store your Quartus II project files for this design, for example, `c:\altera\my_first_fpga`.

   ☞ 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.

Figure 1–3. Project Information



New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

c:\altera\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_top

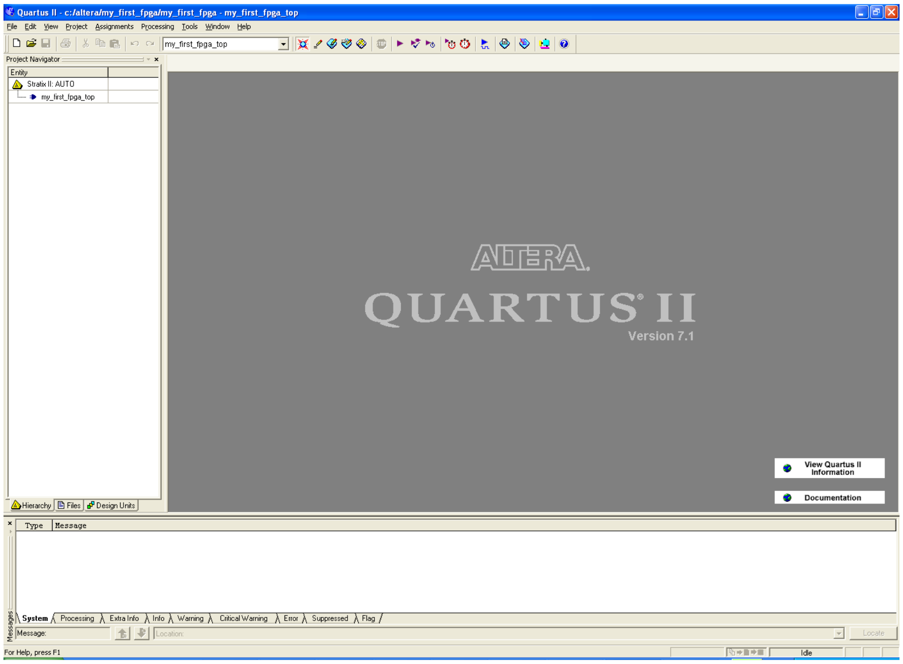
d. Click **Finish.**

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


Quartus II - c:/altera/my_first_fpga/my_first_fpga - my_first_fpga_top
File Edit View Project Assignments Processing Tools Window Help
my_first_fpga_top
Project Navigator
Entity
Stratix II: AUTO
my_first_fpga_top
Hierarchy Files Design Units
ALTERA
QUARTUS II
Version 7.1
View Quartus II Information
Documentation
Type Message
System Processing Extra Info Info Warning Critical Warning Error Suppressed Flag
Messages
Message:
Location:
Locate
For Help, press F1
Idle

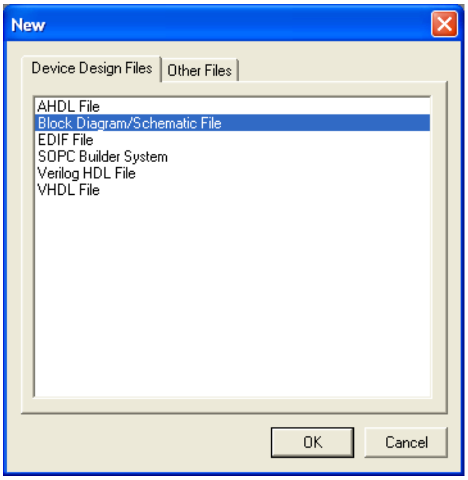
Design Entry

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.

1. Choose **File > New > Block Diagram/Schematic File** (see Figure 1–5) to create a new file, **Block1.bdf**, which you will save as the top-level design.

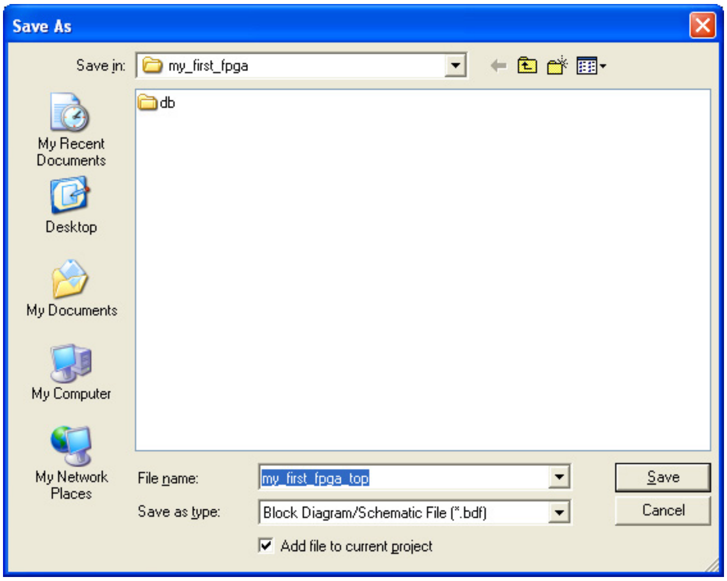
Figure 1–5. New BDF



2. Choose **File > Save As** and enter the following information (see Figure 1–6).

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

Figure 1–6. Saving the BDF



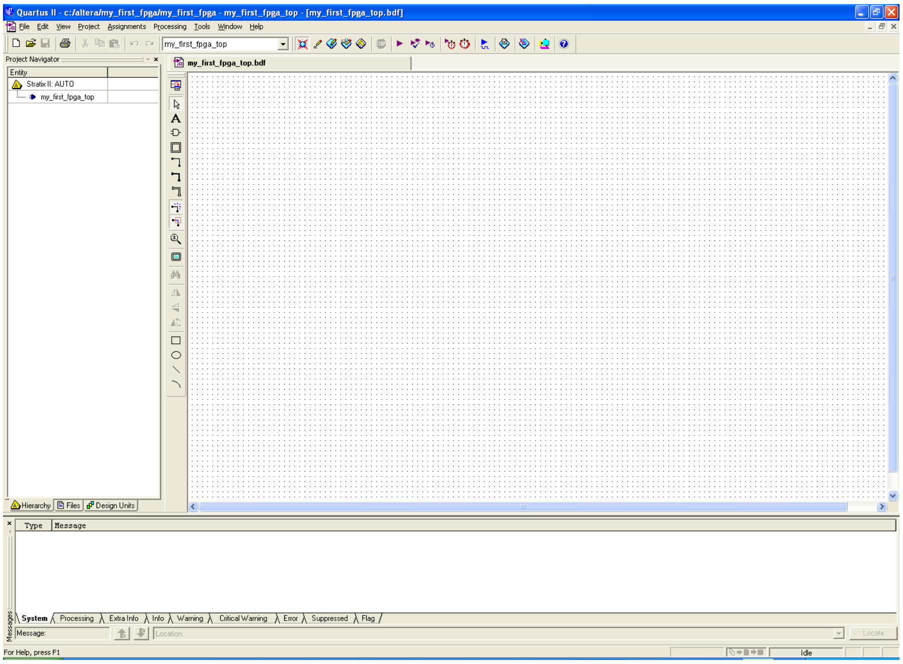
File name: my_first_fpga_top

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

☑ Add file to current project

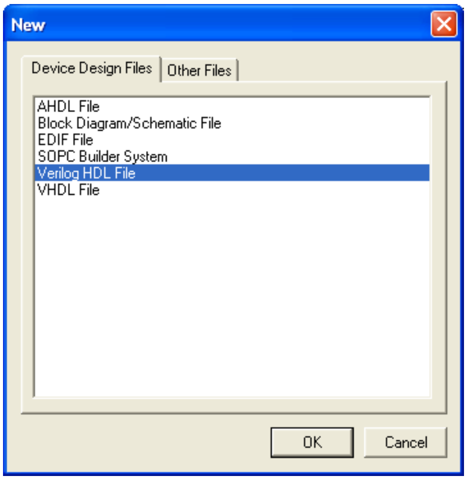
3. Click **Save.** The new design file appears in the Block Editor (see Figure 1–7).

Figure 1–7. Blank BDF



4. Add HDL code to the blank block diagram by choosing **File > New > Verilog HDL File** (see Figure 1–8).

Figure 1–8. New Verilog HDL File

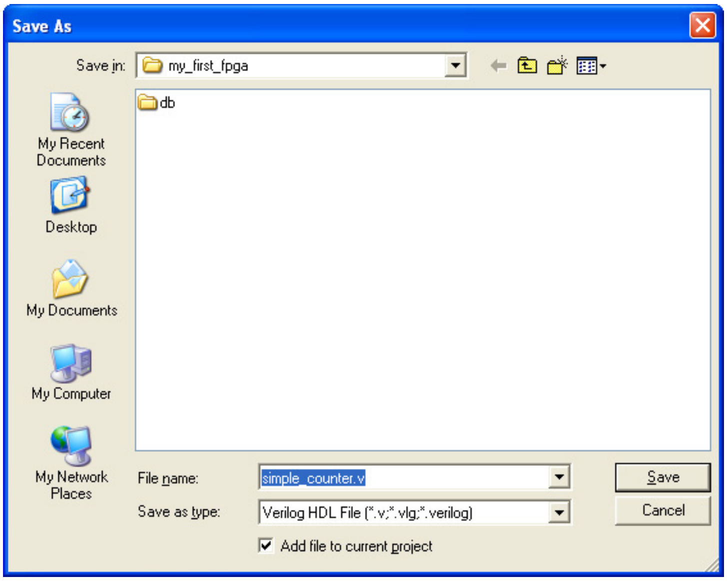


5. Click **OK** to create a new file **Verilog1.v**, which you will save as **simple_counter.v**.

6. 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



File name: simple_counter.v

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

☑ Add file to current project

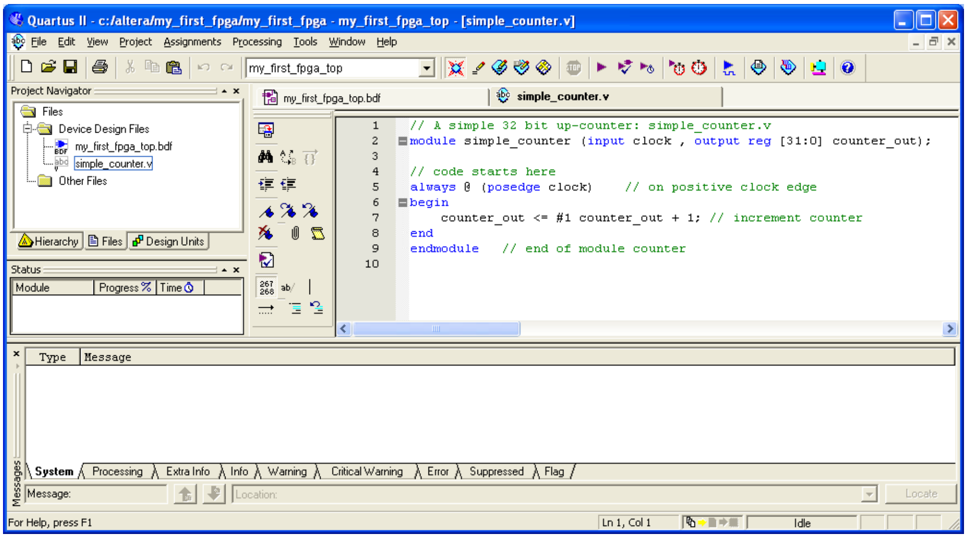
The resulting empty file is ready for you to enter the Verilog HDL code.

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

If you are reading this document as a PDF file, you can copy the code from the PDF and paste it into the blank file.

```
// This is an example of a simple 32 bit up-counter called simple_counter.v
// It has a single clock input and a 32-bit output port
module simple_counter (input clock , output reg [31:0] counter_out);
always @ (posedge clock)// on positive clock edge
begin
 counter_out <= #1 counter_out + 1;// increment counter
end
endmodule// end of module counter
```

Figure 1–10. simple_counter.v



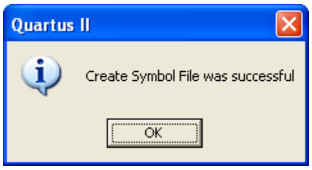
```
// A simple 32 bit up-counter: simple_counter.v
module simple_counter (input clock , output reg [31:0] counter_out);

// code starts here
always @ (posedge clock)    // on positive clock edge
begin
    counter_out <= #1 counter_out + 1; // increment counter
end
endmodule   // end of module counter
```

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

9. Choose **File > Create/Update > Create Symbol Files for Current Update** 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

Create Symbol File was successful

10. Click **OK**.

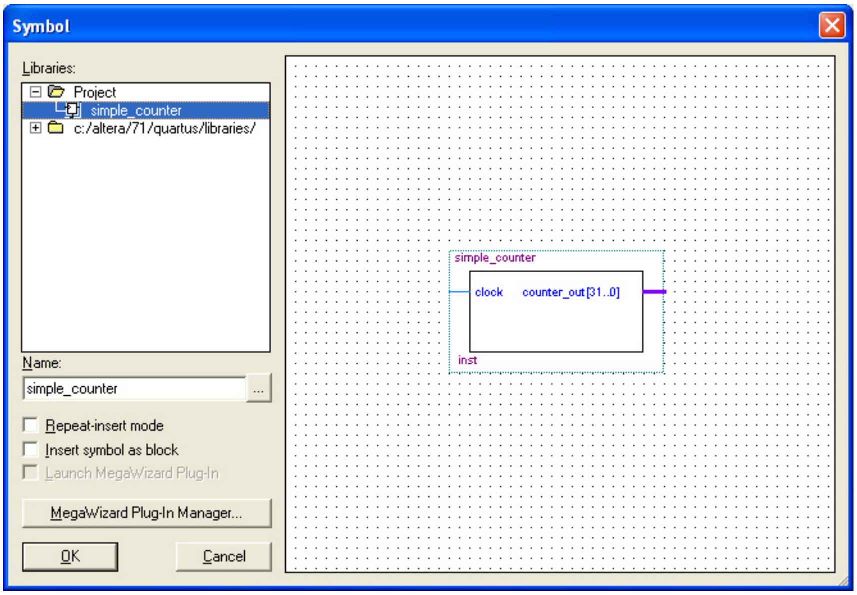
11. To add the **simple_counter.v** symbol to the top-level design, click the **my_first_fpga_top.bdf** tab.

12. Choose **Edit > Insert Symbol**.

13. Double-click the project directory, which shows the newly created `simple_counter` symbol, and select it by clicking it's icon.

    You can also double-click in a blank area of the BDF to open the **Symbol** dialog box

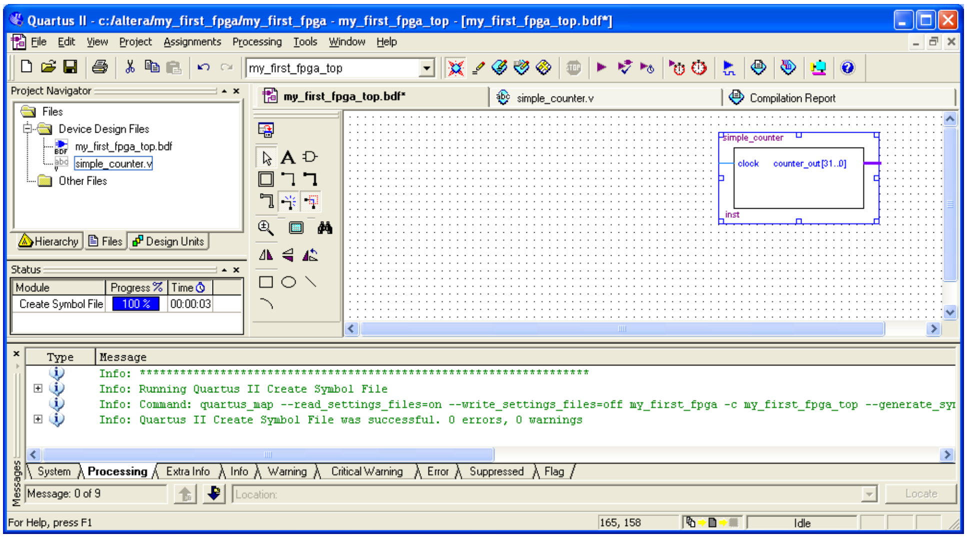
Figure 1–12. Adding the Symbol to the BDF



14. Click **OK**.

15. 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



Module	Progress %	Time
Create Symbol File	100 %	00:00:03

```
Type  Message
      Info: *********************************************************************
      Info: Running Quartus II Create Symbol File
      Info: Command: quartus_map --read_settings_files=on --write_settings_files=off my_first_fpga -c my_first_fpga_top --generate_sym
      Info: Quartus II Create Symbol File was successful. 0 errors, 0 warnings
```

16. 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

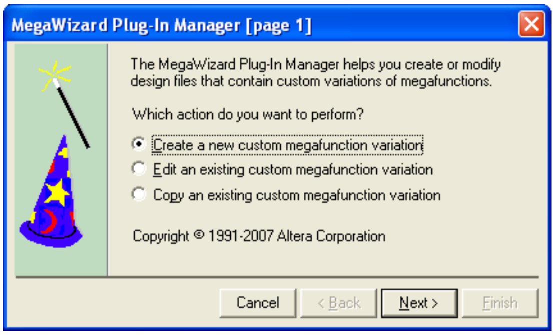
Megafunctions, such as the ones available in the LPM, are pre-designed modules that you can use in FPGA designs. These Altera-provided megafunctions are optimized for speed, area, and device family. You can increase efficiency by using a megafunction instead of writing the function yourself. Altera also 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. To create the clock cource, you will add a pre-built LPM megafunction named ALTPLL.

1. Choose **Edit > Insert Symbol** or click Add Symbol on the toolbar ( ).

2. Click **Megawizard Plug-in Manager**. The MegaWizard® Plug-In Manager appears (see Figure 1–14).

Figure 1–14. MegaWizard Plug-In Manager



MegaWizard 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?

[✓ Create a new custom megafunction variation]

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**.

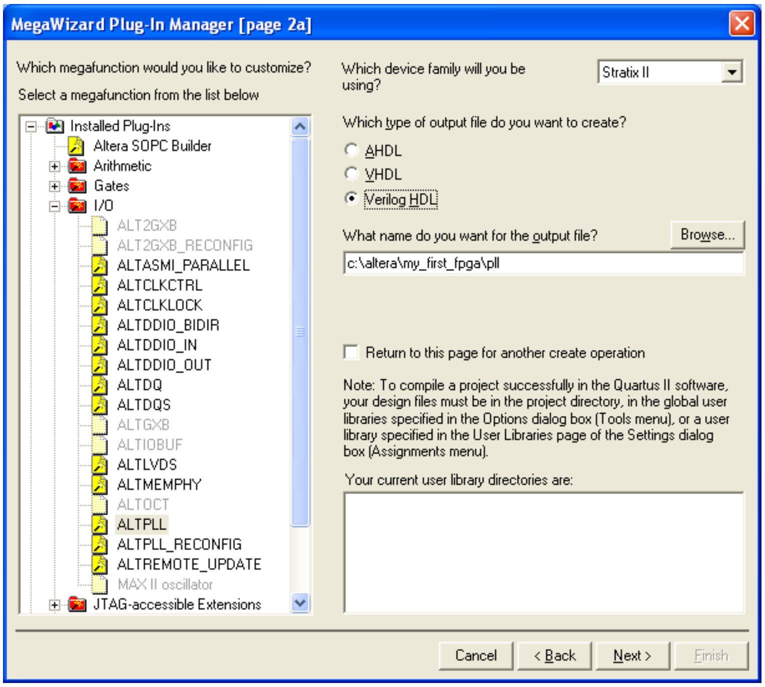
   b. Under **Which device family will you be using?**, choose **Cyclone III**.

   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**.

Figure 1–15. MegaWizard Plug-In Manager [page 2a] Selections


MegaWizard Plug-In Manager [page 2a]

Which megafunction would you like to customize?
Select a megafunction from the list below

- Installed Plug-Ins
  - Altera SOPC Builder
  - Arithmetic
  - Gates
  - I/O
    - ALT2GXB
    - ALT2GXB_RECONFIG
    - ALTASMI_PARALLEL
    - ALTCLKCTRL
    - ALTCLKLOCK
    - ALTDDIO_BIDIR
    - ALTDDIO_IN
    - ALTDDIO_OUT
    - ALTDQ
    - ALTDQS
    - ALTGXB
    - ALTIOBUF
    - ALTLVDS
    - ALTMEMPHY
    - ALTOCT
    - ALTPLL
    - ALTPLL_RECONFIG
    - ALTREMOTE_UPDATE
    - MAX II oscillator
  - JTAG-accessible Extensions

Which device family will you be using? Stratix II

Which type of output file do you want to create?
AHDL
VHDL
[✓ Verilog HDL]

What name do you want for the output file? c:\altera\my_first_fpga\pll

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 user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:

5. In the **MegaWizard Plug-In Manager [page 3 of 12]** window, make the following selections (see Figure 1–16):

   a. Confirm that **General > Which device family will you be using?** is **Cyclone III**.

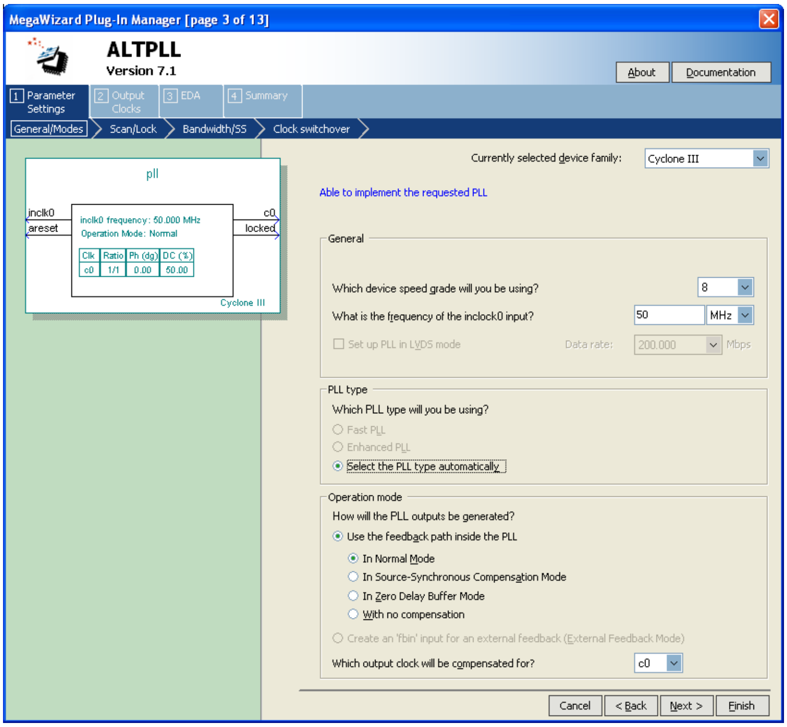
   b. Under **Which device speed grade will you be using?**, type `8`.

   c. Under **What is the frequency of the inclock0 input?**, type `50`.

   d. Ensure that the units are **MHz** (default).

   e. Click **Next**.

Figure 1–16. MegaWizard Plug-In Manager [page 3 of 12] Selections



MegaWizard Plug-In Manager [page 3 of 13]

ALTPLL
Version 7.1

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Currently selected device family: Cyclone III

Able to implement the requested PLL

General

Which device speed grade will you be using? 8

What is the frequency of the inclock0 input? 50 MHz

☐ Set up PLL in LVDS mode Data rate: 200.000 Mbps

PLL type

Which PLL type will you be using?

○ Fast PLL
○ Enhanced PLL
● Select the PLL type automatically

Operation mode

How will the PLL outputs be generated?

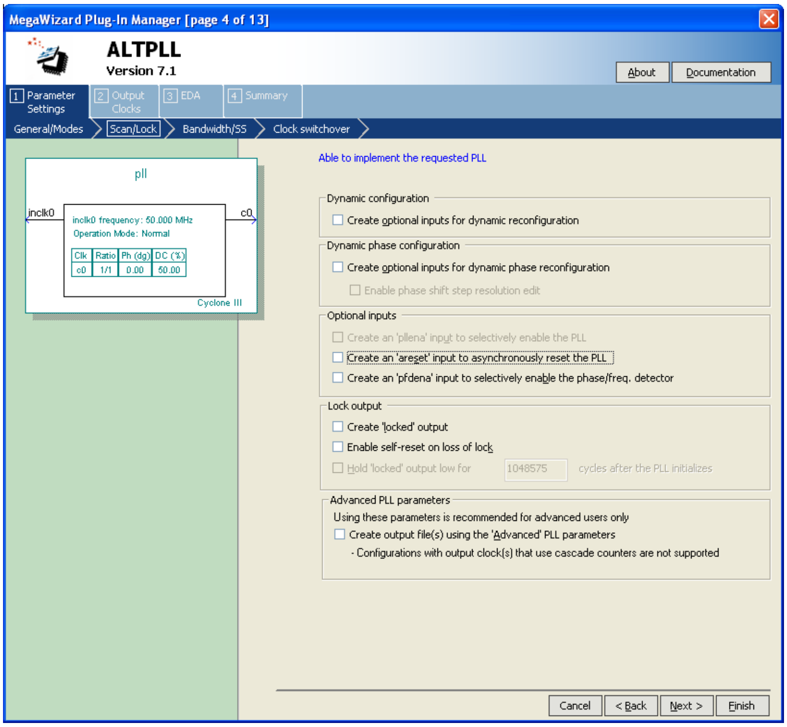
● Use the feedback path inside the PLL
- ● In Normal Mode
- ○ In Source-Synchronous Compensation Mode
- ○ In Zero Delay Buffer Mode
- ○ With no compensation

○ Create an 'fbin' input for an external feedback (External Feedback Mode)

Which output clock will be compensated for? c0

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.

Figure 1–17. MegaWizard Plug-In Manager [page 4 of 12] Selections



inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Able to implement the requested PLL

Dynamic configuration
- ☐ Create optional inputs for dynamic reconfiguration

Dynamic phase configuration
- ☐ Create optional inputs for dynamic phase reconfiguration
  - ☐ Enable phase shift step resolution edit

Optional inputs
- ☐ Create an 'pllena' input to selectively enable the PLL
- ☐ Create an 'areset' input to asynchronously reset the PLL
- ☐ Create an 'pfdena' input to selectively enable the phase/freq. detector

Lock output
- ☐ Create 'locked' output
- ☐ Enable self-reset on loss of lock
- ☐ Hold 'locked' output low for 1048575 cycles after the PLL initializes

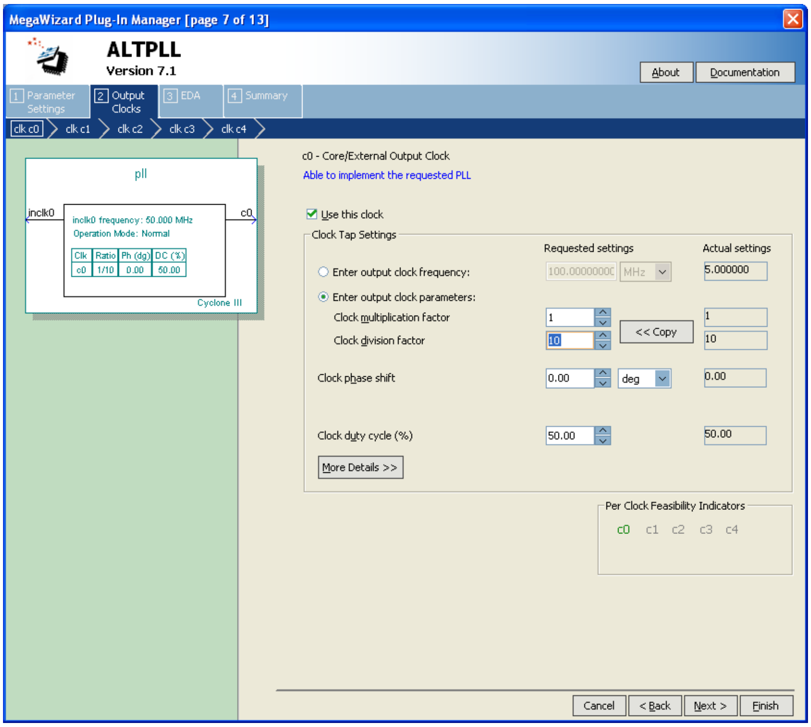
Advanced PLL parameters
Using these parameters is recommended for advanced users only
- ☐ Create output file(s) using the 'Advanced' PLL parameters
  - Configurations with output clock(s) that use cascade counters are not supported

7. Click **Next.**

8. At the top of the wizard, click the tab **2. Output Clocks** to jump to page 7.

9. Under **Clock division factor**, use the up/down arrows or enter `10`. See Figure 1–18.

Figure 1–18. MegaWizard Plug-In Manager [page 7 of 12] Selections


MegaWizard Plug-In Manager [page 7 of 13]

ALTPLL
Version 7.1

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/10	0.00	50.00

c0 - Core/External Output Clock
Able to implement the requested PLL

[✓] Use this clock

Clock Tap Settings

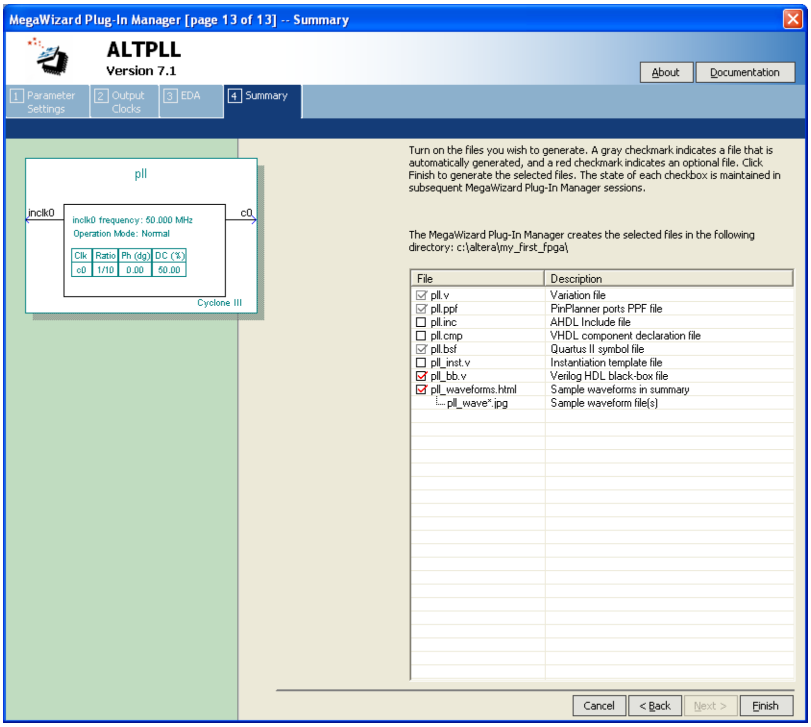
	Requested settings		Actual settings
Enter output clock frequency:	100.0000000	MHz	5.000000
[● Enter output clock parameters:]			
Clock multiplication factor	1		1
Clock division factor	10		10
Clock phase shift	0.00	deg	0.00
Clock duty cycle (%)	50.00		50.00

Per Clock Feasibility Indicators
c0 c1 c2 c3 c4

10. Click **Finish**.

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

Figure 1–19. Wizard-Created Files



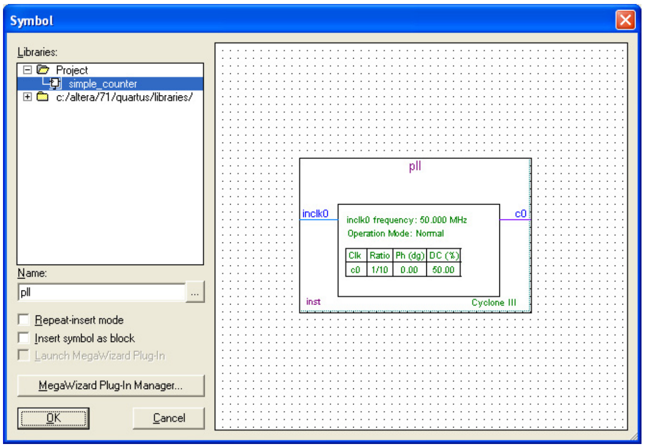
Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

The MegaWizard Plug-In Manager creates the selected files in the following directory: c:\altera\my_first_fpga\

File	Description
✓ pll.v	Variation file
✓ pll.ppf	PinPlanner ports PPF file
☐ pll.inc	AHDL Include file
☐ pll.cmp	VHDL component declaration file
✓ pll.bsf	Quartus II symbol file
☐ pll_inst.v	Instantiation template file
✓ pll_bb.v	Verilog HDL black-box file
✓ pll_waveforms.html	Sample waveforms in summary
pll_wave*.jpg	Sample waveform file(s)

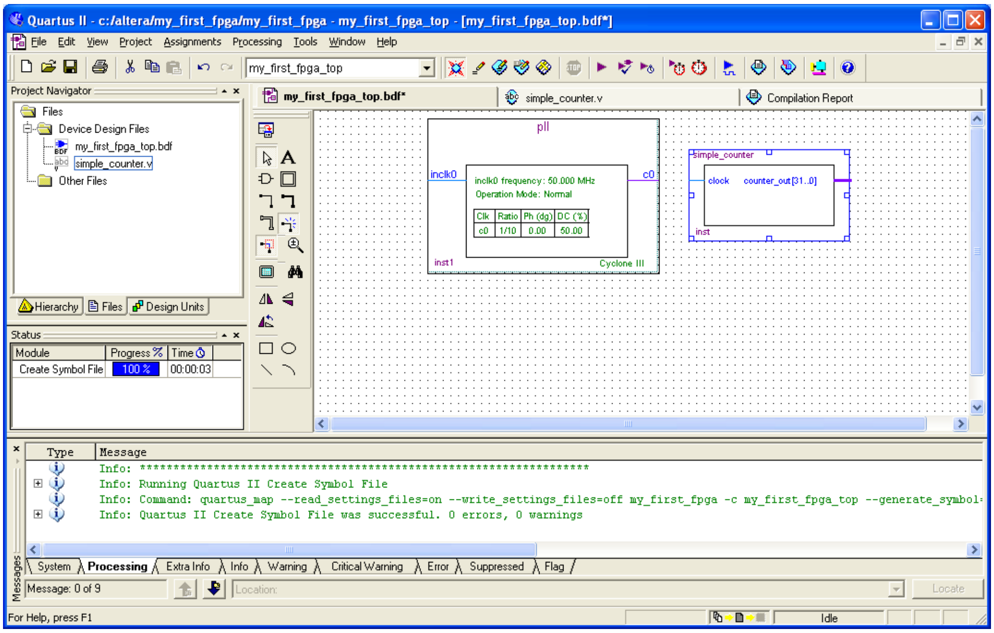
The Symbol window opens, showing the newly created PLL megafunction. See Figure 1–20.

Figure 1–20. pll Symbol



12. Click **OK** and place the `pll` symbol onto the BDF to the left of the `simple_counter` symbol. See Figure 1–21.

Figure 1–21. Place the pll Symbol

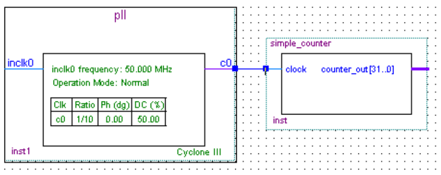


```
Type    Message
        Info: ***********************************************************************
        Info: Running Quartus II Create Symbol File
        Info: Command: quartus_map --read_settings_files=on --write_settings_files=off my_first_fpga -c my_first_fpga_top --generate_symbol=
        Info: Quartus II Create Symbol File was successful. 0 errors, 0 warnings
```

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

14. 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).

Figure 1–22. Draw a Bus Line from pll to simple_counter



inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/10	0.00	50.00

15. Add an input pin and an output bus with the following steps:

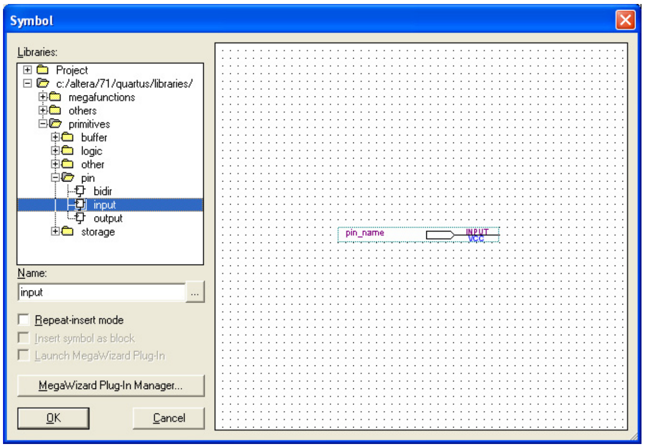
    a. Choose **Edit > Insert Symbol**.

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

    c. Click **OK**. See Figure 1–23.

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.

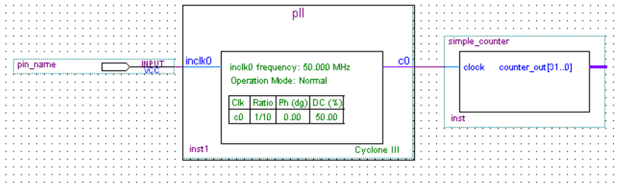
Figure 1–23. Input Pin Symbol



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

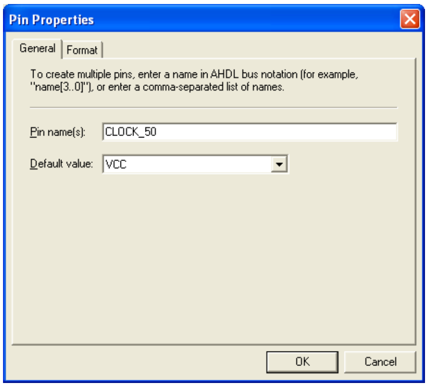
e. Use the mouse to click and drag the new input pin to the left; notice that the ports remain connected as shown in Figure 1–24.

Figure 1–24. Connecting Input Pin



f. Change the pin name by double-clicking `pin_name` and typing `CLOCK_50` (see Figure 1–25). This pin corresponds to the development board's `CLOCK_50` input pin, which is connected to the Cyclone III FPGA on the board.

Figure 1–25. Change Input Pin Name



To create multiple pins, enter a name in AHDL bus notation (for example, "name[3..0]"), or enter a comma-separated list of names.

Pin name(s): CLOCK_50

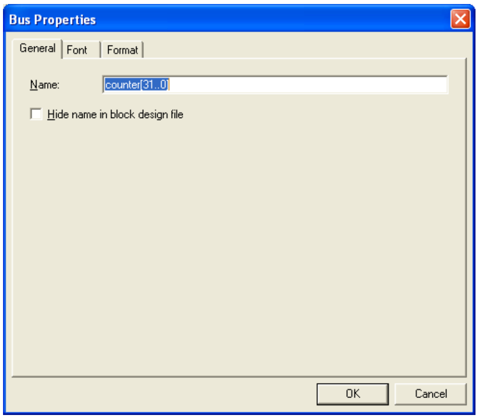
Default value: VCC

g. Using the Orthoganal Bus tool, draw a bus line from the `simple_counter` output port.

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

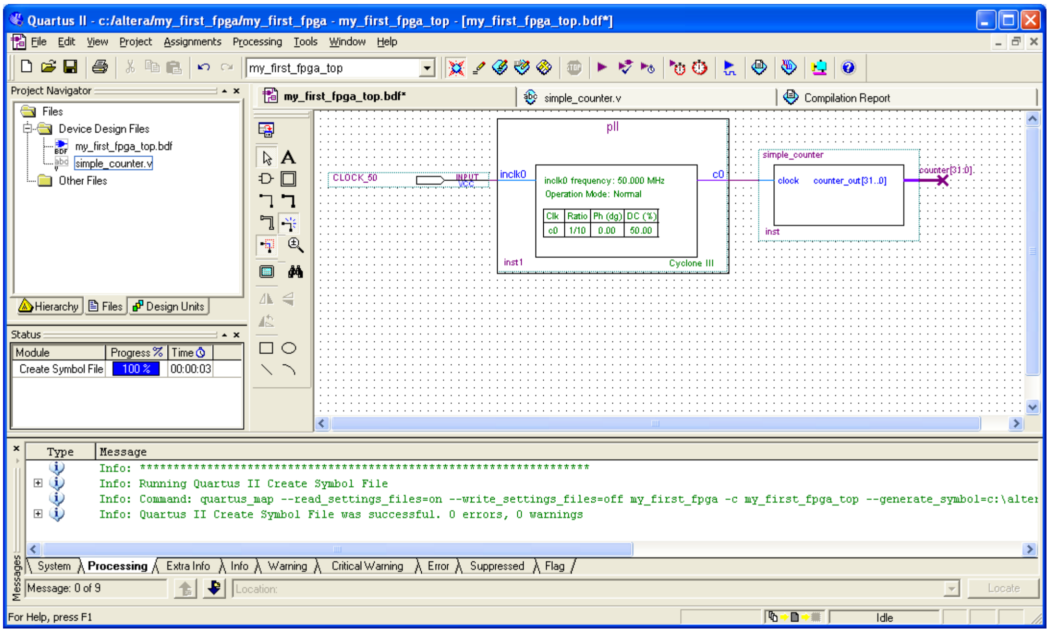


Name: counter[31..0]

Hide name in block design file

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

Figure 1–27. BDF



Status

Module	Progress %	Time
Create Symbol File	100 %	00:00:03

```
Type  Message
      Info: *******************************************************************
      Info: Running Quartus II Create Symbol File
      Info: Command: quartus_map --read_settings_files=on --write_settings_files=off my_first_fpga -c my_first_fpga_top --generate_symbol=c:\alter
      Info: Quartus II Create Symbol File was successful. 0 errors, 0 warnings
```

System | Processing | Extra Info | Info | Warning | Critical Warning | Error | Suppressed | Flag

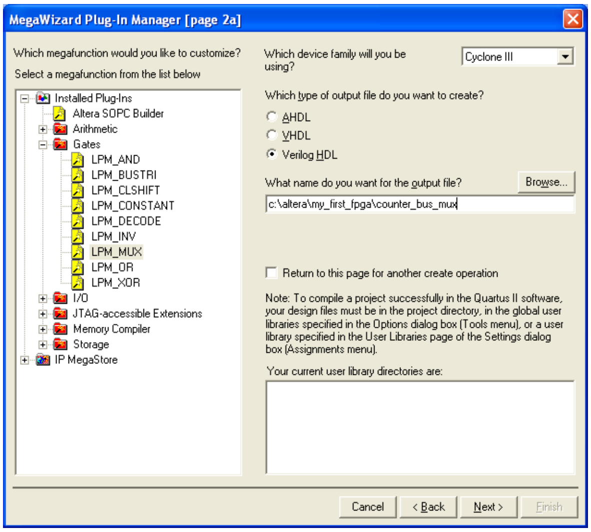
Message: 0 of 9   Location: Locate

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 **Cyclone III** as the device family and name the output file **counter_bus_mux** (see Figure 1–28).
6. Click **Next**.

Figure 1–28. Selecing lpm_mux


MegaWizard Plug-In Manager [page 2a]

Which megafunction would you like to customize?
Select a megafunction from the list below

- Installed Plug-Ins
  - Altera SOPC Builder
  - Arithmetic
  - Gates
    - LPM_AND
    - LPM_BUSTRI
    - LPM_CLSHIFT
    - LPM_CONSTANT
    - LPM_DECODE
    - LPM_INV
    - LPM_MUX
    - LPM_OR
    - LPM_XOR
  - I/O
  - JTAG-accessible Extensions
  - Memory Compiler
  - Storage
- IP MegaStore

Which device family will you be using? Cyclone III

Which type of output file do you want to create?
AHDL
VHDL
[✓ Verilog HDL]

What name do you want for the output file? c:\altera\my_first_fpga\counter_bus_mux

Return to this page for another create operation

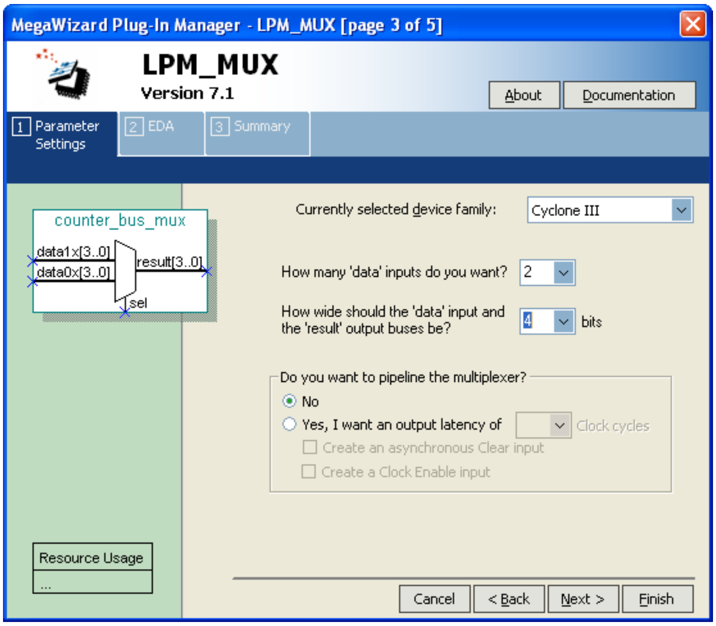
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:

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).

Figure 1–29. lpm_mux Settings


MegaWizard Plug-In Manager - LPM_MUX [page 3 of 5]

LPM_MUX
Version 7.1

Currently selected device family: Cyclone III

How many 'data' inputs do you want? 2

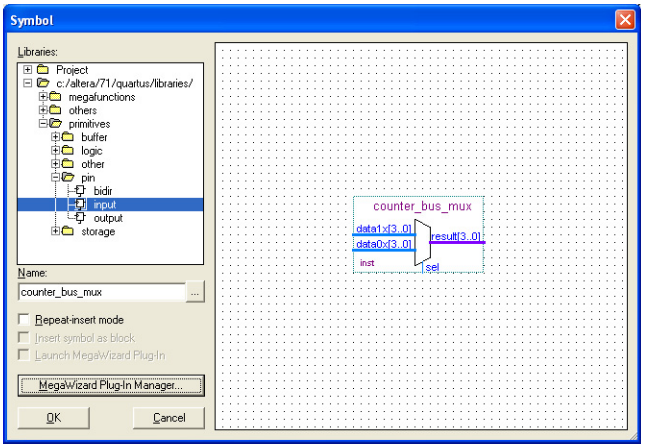
How wide should the 'data' input and the 'result' output buses be? 4 bits

Do you 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

9. Click **Next**.

10. Click **Finish**. The Symbol window appears (see Figure 1–30).

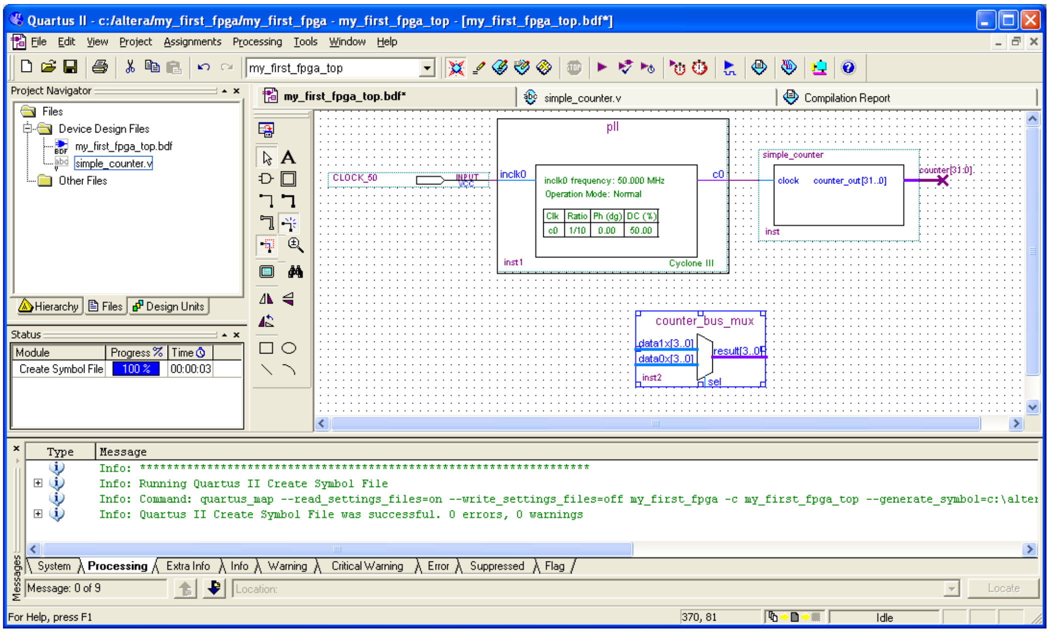
Figure 1–30. lpm_mux Symbol



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



Module	Progress %	Time
Create Symbol File	100 %	00:00:03

```
Type  Message
      Info: ***************************************************************************
      Info: Running Quartus II Create Symbol File
      Info: Command: quartus_map --read_settings_files=on --write_settings_files=off my_first_fpga -c my_first_fpga_top --generate_symbol=c:\alter
      Info: Quartus II Create Symbol File was successful. 0 errors, 0 warnings
```

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

    a. Using the Orthoganal Bus tool, draw bus lines from the `data1x[3..0]` and `data0x[3..0]` input ports.

    b. Draw a bus line from the `result[3..0]` output port.

    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 for this input bus.

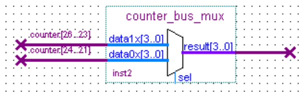
    e. Click **OK**.

    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 for this input bus.

h. Click **OK**. Figure 1–32 shows the renamed buses.

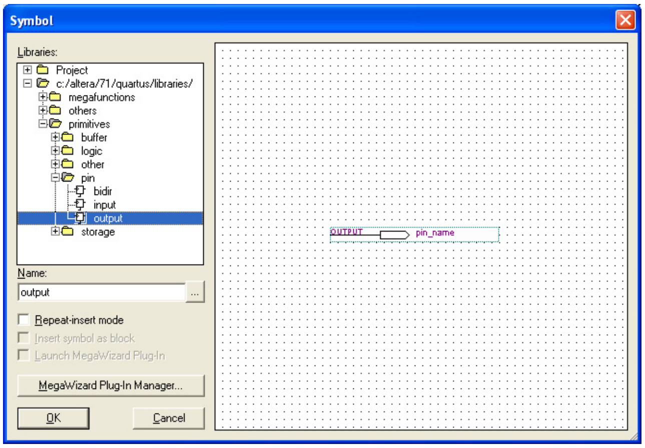
Figure 1–32. Renamed counter_bus_mux Bus Lines



14. Choose **Edit > Insert Symbol**.

15. Under **Libraries**, double-click **quartus/libraries/ directory > primitives > pin > output** (see Figure 1–33).

Figure 1–33. Choose an Output Pin

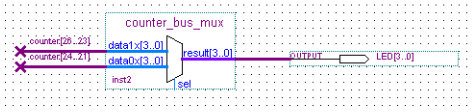


16. Click **OK**.

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. With this name, it corresponds to the output pins `LED0` through `LED3` on the board, which are connected to the Cyclone III FPGA (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**.

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

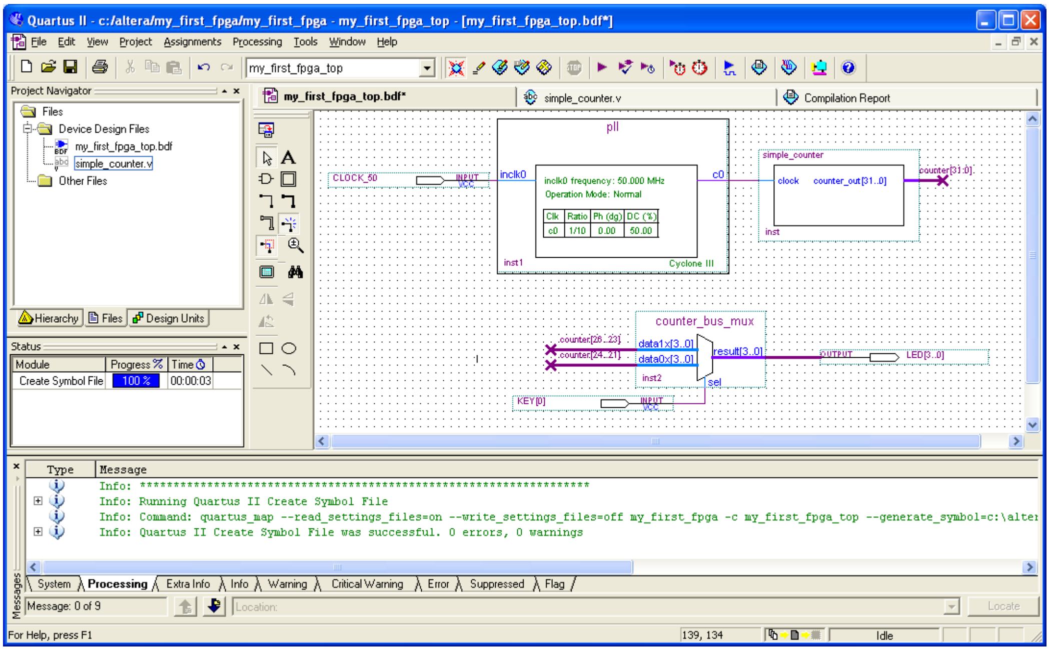
    c. Click **OK**.

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 `KEY[0]` (see Figure 1–35).

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



Module	Progress %	Time
Create Symbol File	100 %	00:00:03

```
Type   Message
       Info: ********************************************************************
       Info: Running Quartus II Create Symbol File
       Info: Command: quartus_map --read_settings_files=on --write_settings_files=off my_first_fpga -c my_first_fpga_top --generate_symbol=c:\alter
       Info: Quartus II Create Symbol File was successful. 0 errors, 0 warnings
```

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 `KEY[0]` input pin and add a project description, such as "My First FPGA Project."

Assign Cyclone III Device and Pins

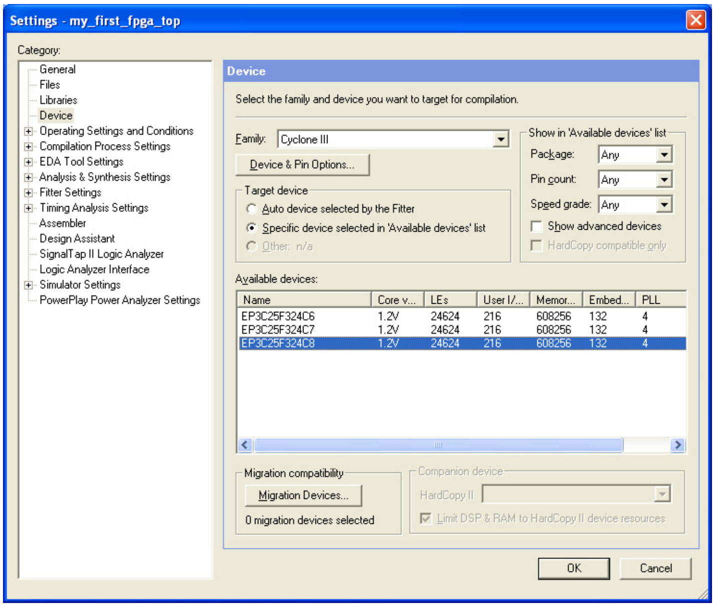
In this section, you will assign a specific Cyclone III FPGA device to the design and make pin assignments. To assign the device, perform the following steps.

1. Choose **Assignments > Device**.

2. Under **Family**, choose **Cyclone III**.

3. Under **Available devices**, choose **EP3C25F324C8** as the device. See Figure 1–36.

Figure 1–36. Specify the Cyclone III Device



Settings - my_first_fpga_top

Category:

- General
- Files
- Libraries
- Device
- Operating Settings and Conditions
- Compilation Process Settings
- EDA Tool Settings
- Analysis & Synthesis Settings
- Fitter Settings
- Timing Analysis Settings
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- Simulator Settings
- PowerPlay Power Analyzer Settings

Device

Select the family and device you want to target for compilation.

Family: Cyclone III

Show in 'Available devices' list
Package: Any
Pin count: Any
Speed grade: Any
☐ Show advanced devices
☐ HardCopy compatible only

Target device
○ Auto device selected by the Fitter
● Specific device selected in 'Available devices' list
○ Other: n/a

Available devices:

Name	Core v...	LEs	User I/...	Memor...	Embed...	PLL
EP3C25F324C6	1.2V	24624	216	608256	132	4
EP3C25F324C7	1.2V	24624	216	608256	132	4
EP3C25F324C8	1.2V	24624	216	608256	132	4

Migration compatibility
0 migration devices selected

Companion device
HardCopy II
☑ Limit DSP & RAM to HardCopy II device resources

4. Click **OK**.

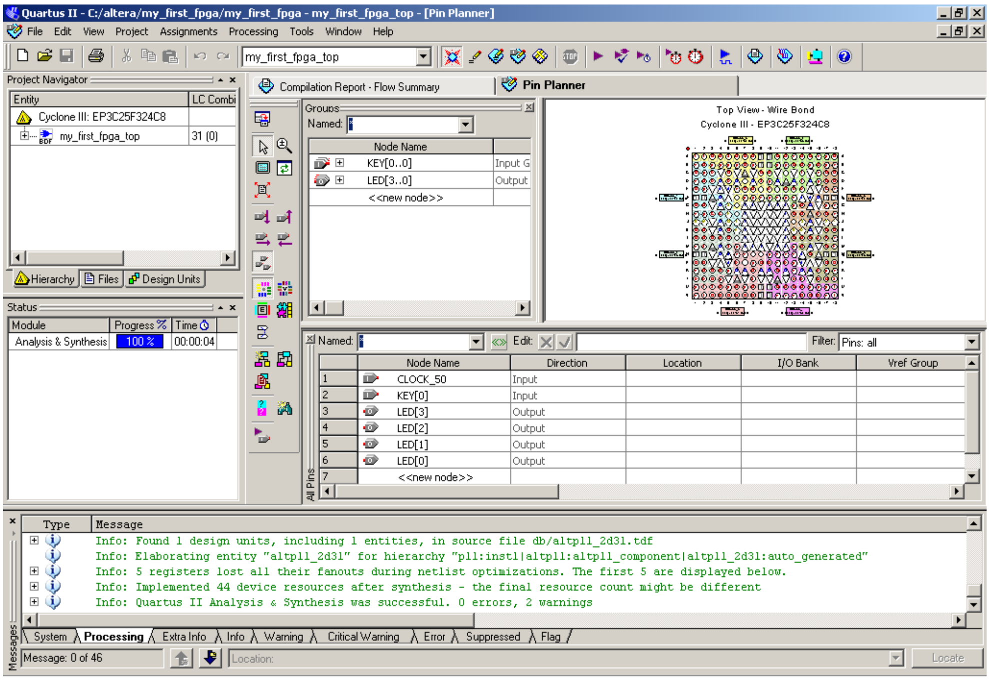
5. Choose **Processing > Start > Start Analysis & Elaboration** in preparation for assigning pin locations.

6. 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 1–37.

Figure 1–37. Pin Planner


Quartus II - C:/altera/my_first_fpga/my_first_fpga - my_first_fpga_top - [Pin Planner]

Entity	LC Combi
Cyclone III: EP3C25F324C8	
my_first_fpga_top	31 (0)

Module	Progress %	Time
Analysis & Synthesis	100 %	00:00:04

Node Name	
KEY[0..0]	Input G
LED[3..0]	Output
<<new node>>	

Top View - Wire Bond
Cyclone III - EP3C25F324C8

Filter: Pins: all

	Node Name	Direction	Location	I/O Bank	Vref Group
1	CLOCK_50	Input			
2	KEY[0]	Input			
3	LED[3]	Output			
4	LED[2]	Output			
5	LED[1]	Output			
6	LED[0]	Output			
7	<<new node>>				

```
Type  Message
      Info: Found 1 design units, including 1 entities, in source file db/altpll_2d31.tdf
      Info: Elaborating entity "altpll_2d31" for hierarchy "pll:inst1|altpll:altpll_component|altpll_2d31:auto_generated"
      Info: 5 registers lost all their fanouts during netlist optimizations. The first 5 are displayed below.
      Info: Implemented 44 device resources after synthesis - the final resource count might be different
      Info: Quartus II Analysis & Synthesis was successful. 0 errors, 2 warnings
```

Message: 0 of 46

2. In the **Location** column next to each of the six node names, add the FineLine® BGA (FBGA) coordinates (pin numbers) as shown in Table 1–1.

Table 1–1. Location Pin Settings

Row	Node Name	Location
1	`KEY[0]`	F1
2	`CLOCK_50`	V9
3	`LED[0]`	P13
4	`LED[1]`	P12
5	`LED[2]`	N12
6	`LED[3]`	N9

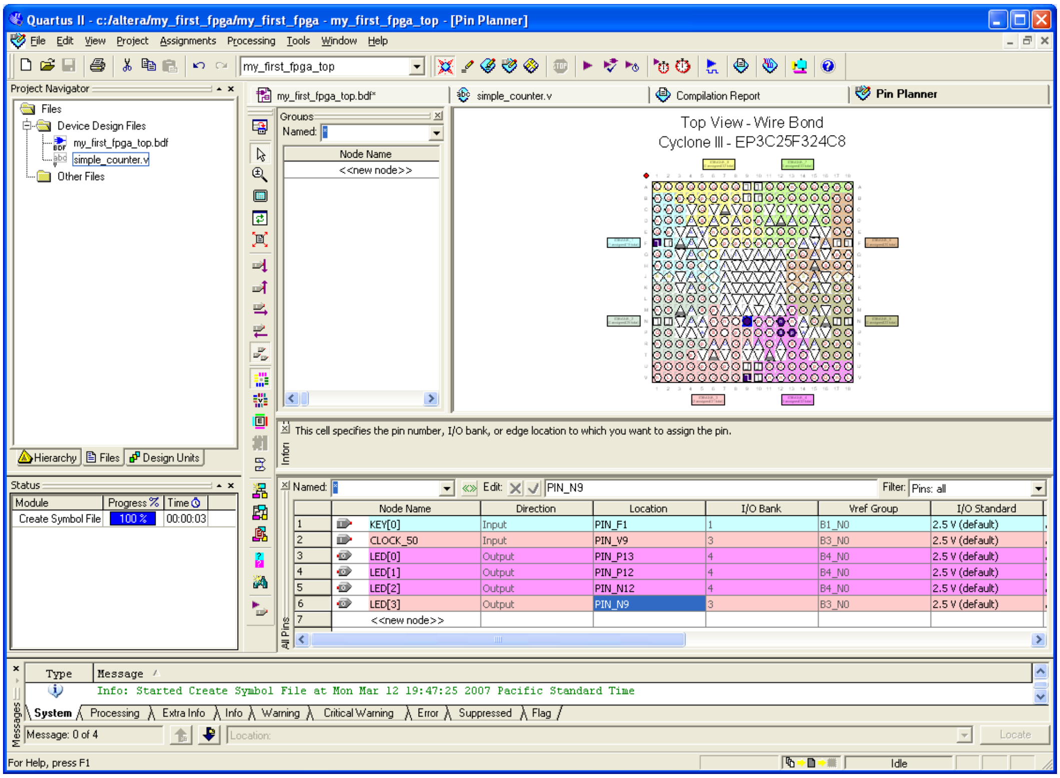
Double-click in the **Location** column for any of the six pins to open a drop-down list. You can select the pin from this list. Alternatively, you can type the location. For example, if you type `F1` and press the Enter key, the Quartus II software fills in the correct `PIN_F1` location name for you. 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 1–38.

☞ As a shortcut, drag and drop the pin from the FPGA pin diagram into the pin table. Make sure that you select the correct pins when using this method.

For Cyclone III devices, unused I/O pins default to tri-stated inputs. You can change this setting using the **Unused Pins** tab in the **Device and Pin Options** dialog box. Refer to Quartus II Help for more information about this option and how to use it.

☞ 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.

Figure 1–38. Completed Pin Planning



Module	Progress %	Time
Create Symbol File	100 %	00:00:03

Node Name
<<new node>>

This cell specifies the pin number, I/O bank, or edge location to which you want to assign the pin.

Edit: PIN_N9

	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard
1	KEY[0]	Input	PIN_F1	1	B1_N0	2.5 V (default)
2	CLOCK_50	Input	PIN_V9	3	B3_N0	2.5 V (default)
3	LED[0]	Output	PIN_P13	4	B4_N0	2.5 V (default)
4	LED[1]	Output	PIN_P12	4	B4_N0	2.5 V (default)
5	LED[2]	Output	PIN_N12	4	B4_N0	2.5 V (default)
6	LED[3]	Output	PIN_N9	3	B3_N0	2.5 V (default)
7	<<new node>>					

Type	Message
i	Info: Started Create Symbol File at Mon Mar 12 19:47:25 2007 Pacific Standard Time

You are finished creating your Quartus II design!

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 is 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.

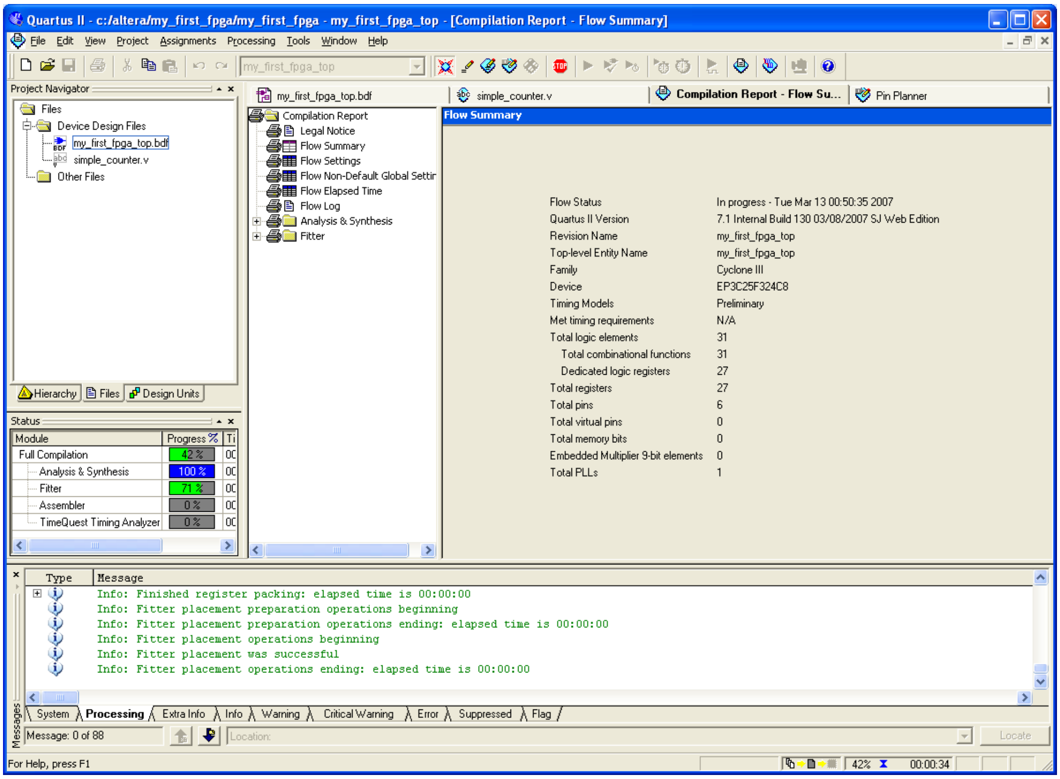
Now that you have created a complete Quartus II project and entered all assignments, you can compile the design.

✓ 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 Messages



Status

Module	Progress %	Ti
Full Compilation	42 %	0C
Analysis & Synthesis	100 %	0C
Fitter	71 %	0C
Assembler	0 %	0C
TimeQuest Timing Analyzer	0 %	0C

Flow Status	In progress - Tue Mar 13 00:50:35 2007
Quartus II Version	7.1 Internal Build 130 03/08/2007 SJ Web Edition
Revision Name	my_first_fpga_top
Top-level Entity Name	my_first_fpga_top
Family	Cyclone III
Device	EP3C25F324C8
Timing Models	Preliminary
Met timing requirements	N/A
Total logic elements	31
Total combinational functions	31
Dedicated logic registers	27
Total registers	27
Total pins	6
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	1

```
Type  Message
      Info: Finished register packing: elapsed time is 00:00:00
      Info: Fitter placement preparation operations beginning
      Info: Fitter placement preparation operations ending: elapsed time is 00:00:00
      Info: Fitter placement operations beginning
      Info: Fitter placement was successful
      Info: Fitter placement operations ending: elapsed time is 00:00:00
```

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

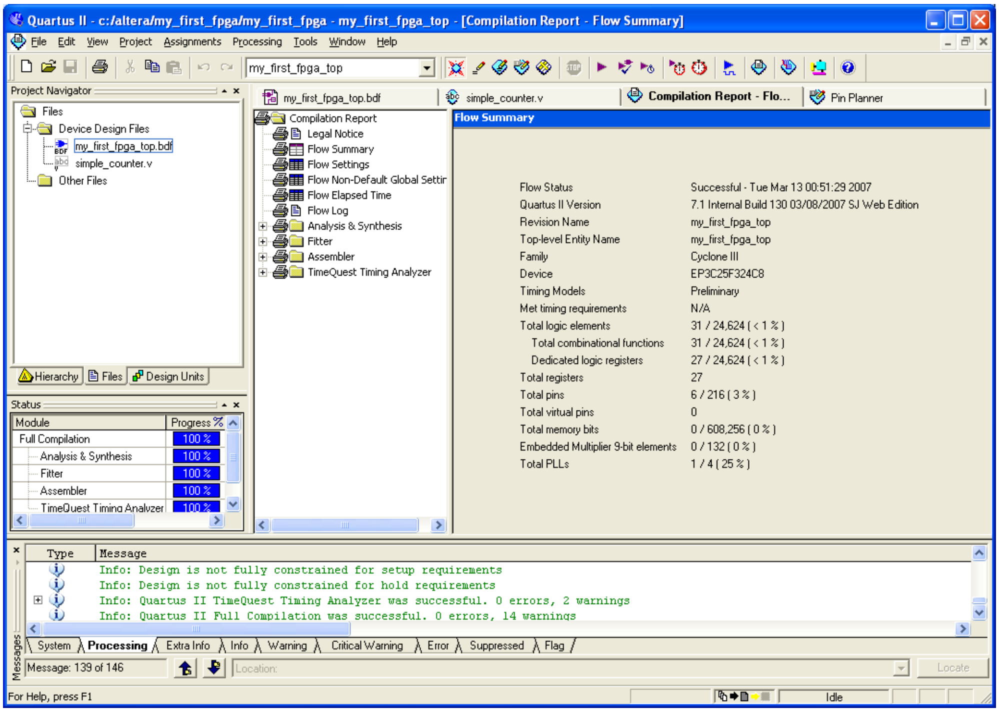
The Quartus II Messages window displays many messages during compilation. In this exercise, you did not create any timing constraints. Therefore, the Quartus II software issues two critical warnings:

Critical Warning: SDC file not found: 'my_first_fpga_top.sdc'
Critical Warning: Timing requirements not met

The second warning is displayed because no timing requirements were made, therefore, they could not be met. For this tutorial project, you can ignore these warnings. However, setting timing requirements is a critical part of creating successful designs. For more information on setting timing requirements, refer to Quartus II Help and/or the Altera etraining courses mentioned in "Next Steps" on page 1–43.

The software provides the compilation results in the **Compilation Report** tab as shown in Figure 1–40.

Figure 1–40. Compilation Report


Quartus II - c:/altera/my_first_fpga/my_first_fpga - my_first_fpga_top - [Compilation Report - Flow Summary]

Flow Summary

Flow Status	Successful - Tue Mar 13 00:51:29 2007
Quartus II Version	7.1 Internal Build 130 03/08/2007 SJ Web Edition
Revision Name	my_first_fpga_top
Top-level Entity Name	my_first_fpga_top
Family	Cyclone III
Device	EP3C25F324C8
Timing Models	Preliminary
Met timing requirements	N/A
Total logic elements	31 / 24,624 (< 1 %)
Total combinational functions	31 / 24,624 (< 1 %)
Dedicated logic registers	27 / 24,624 (< 1 %)
Total registers	27
Total pins	6 / 216 (3 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

Module	Progress %
Full Compilation	100 %
Analysis & Synthesis	100 %
Fitter	100 %
Assembler	100 %
TimeQuest Timing Analyzer	100 %

```
Type  Message
      Info: Design is not fully constrained for setup requirements
      Info: Design is not fully constrained for hold requirements
      Info: Quartus II TimeQuest Timing Analyzer was successful. 0 errors, 2 warnings
      Info: Quartus II Full Compilation was successful. 0 errors, 14 warnings
```

Message: 139 of 146

Program the Device

After compiling and verifying your design you are ready to program the Cyclone III 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. Connect the USB cable to your host computer and the board.

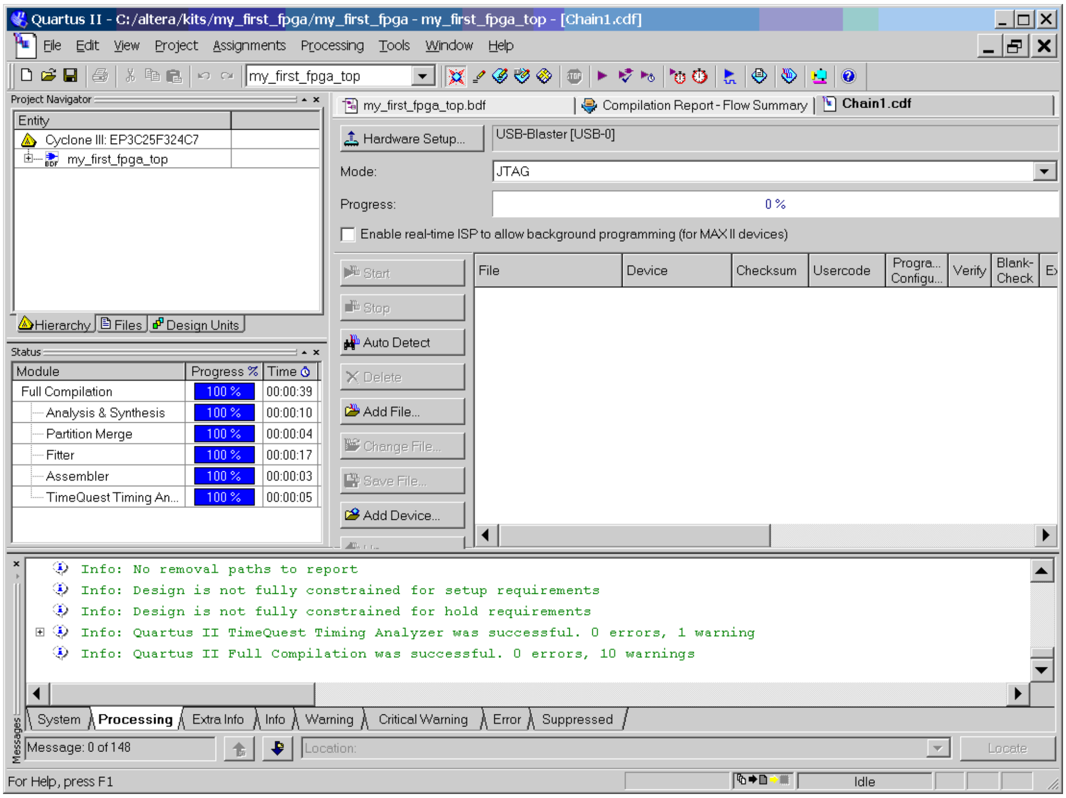
   ☞ Refer to the getting started user guide for detailed instructions on how to connect the cables.

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

Program the FPGA using the following steps.

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

Figure 1–41. Programmer Window



Quartus II - C:/altera/kits/my_first_fpga/my_first_fpga - my_first_fpga_top - [Chain1.cdf]

my_first_fpga_top | my_first_fpga_top.bdf | Compilation Report - Flow Summary | Chain1.cdf

Entity
Cyclone III: EP3C25F324C7
my_first_fpga_top

Hardware Setup... USB-Blaster [USB-0]

Mode: JTAG

Progress: 0 %

Enable real-time ISP to allow background programming (for MAX II devices)

Status

Module	Progress %	Time
Full Compilation	100 %	00:00:39
Analysis & Synthesis	100 %	00:00:10
Partition Merge	100 %	00:00:04
Fitter	100 %	00:00:17
Assembler	100 %	00:00:03
TimeQuest Timing An...	100 %	00:00:05

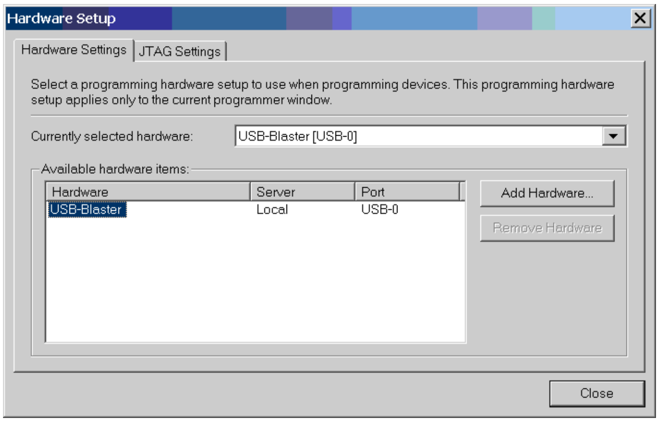
```
Info: No removal paths to report
Info: Design is not fully constrained for setup requirements
Info: Design is not fully constrained for hold requirements
Info: Quartus II TimeQuest Timing Analyzer was successful. 0 errors, 1 warning
Info: Quartus II Full Compilation was successful. 0 errors, 10 warnings
```

Message: 0 of 148

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



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: USB-Blaster [USB-0]

Available hardware items:

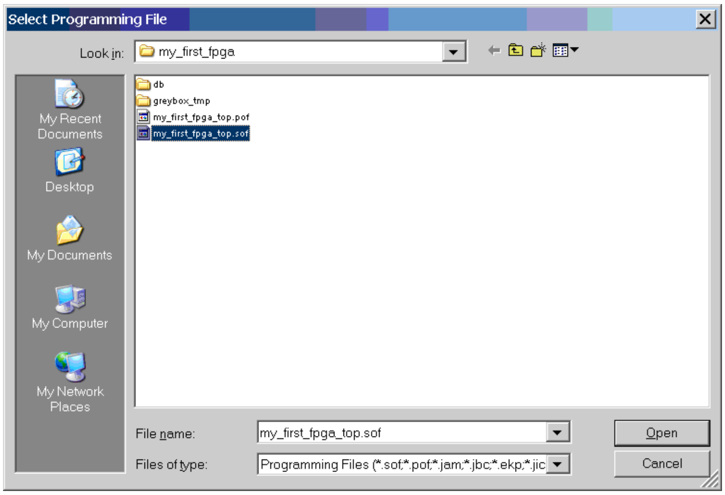
Hardware	Server	Port
USB-Blaster	Local	USB-0

4. Click **Close**.

5. 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



Look in: my_first_fpga

db
greybox_tmp
my_first_fpga_top.pof
my_first_fpga_top.sof

File name: my_first_fpga_top.sof

Files of type: Programming Files (*.sof;*.pof;*.jam;*.jbc;*.ekp;*.jic

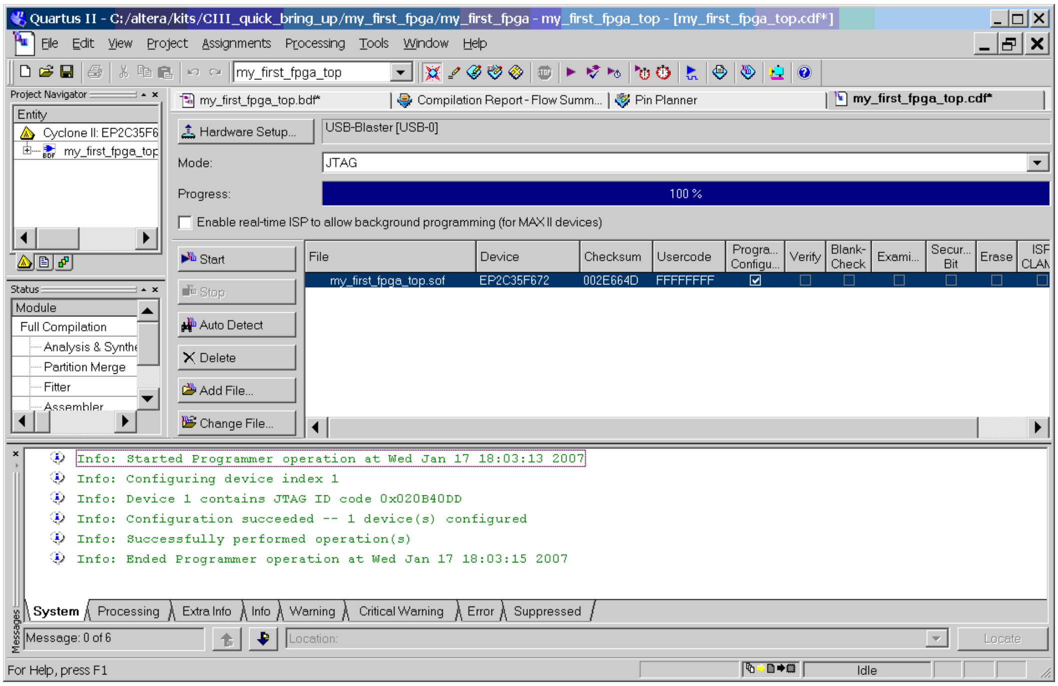
7. Click **Open**.

8. 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.

Figure 1–44. Downloading Complete



USB-Blaster [USB-0]

Mode: JTAG

Progress: 100 %

☐ Enable real-time ISP to allow background programming (for MAX II devices)

File	Device	Checksum	Usercode	Progra... Configu...	Verify	Blank-Check	Exami...	Secur... Bit	Erase	ISP CLAN
my_first_fpga_top.sof	EP2C35F672	002E664D	FFFFFFFF	☑	☐	☐	☐	☐	☐	☐

```
Info: Started Programmer operation at Wed Jan 17 18:03:13 2007
Info: Configuring device index 1
Info: Device 1 contains JTAG ID code 0x020B40DD
Info: Configuration succeeded -- 1 device(s) configured
Info: Successfully performed operation(s)
Info: Ended Programmer operation at Wed Jan 17 18:03:15 2007
```

Congratulations, you have created, compiled, and programmed your first FPGA design! The compiled SRAM Object File (**.sof**) is loaded onto the Cyclone III 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:

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 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 [24..21]).

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: Simulation
- Using the Quartus II Software: Timing Analysis
- Constraining and Analyzing Timing for Source Synchronous Circuits with TimeQuest
- Validating Performance with the TimeQuest Static Timing Analyzer