Quartus™

Programmable Logic Development System

Tutorial

ALTERA®

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

Quartus Tutorial
Version 1999.10 Revision 2
November 1999

P25-04732-01

Altera, the Altera logo, and MAX+PLUS II are registered trademarks of Altera Corporation in the United States and other countries. APEX, APEX 20K, APEX 20KE, Atlas, ByteBlasterMV, MasterBlaster, MegaLAB, MegaWizard, EP20K100, Quartus, and the Quartus logo are trademarks and/or service marks of Altera Corporation in the United States and other countries. Product design elements and mnemonics used by Altera Corporation are protected by copyright and/or trademark laws.

Altera Corporation acknowledges the trademarks of other organizations for their respective products or services mentioned in this document, including the following: Microsoft is a registered trademark and Windows and Windows NT are trademarks of Microsoft Corporation.

Altera reserves the right to make changes, without notice, in the devices or the device specifications identified in this document. Altera advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty. Testing and other quality control techniques are used to the extent Altera deems such testing necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed. In the absence of written agreement to the contrary, Altera assumes no liability for Altera applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does Altera warrant or represent any patent right, copyright, or other intellectual property right of Altera covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Altera products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Altera Corporation. As used herein:

1. Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Products mentioned in this document are covered by one or more of the following U.S. patents: 5,915,017; 5,909,450; 5,909,375; 5,909,126; 5,905,675; 5,904,524; 5,900,743; 5,898,628; 5,898,318; 5,894,228; 5,893,088; 5,892,683; 5,883,526; 5,880,725; 5,880,597; 5,880,596; 5,878,250; 5,875,112; 5,873,113; 5,872,529; 5,872,463; 5,870,410; 5,869,980; 5,869,979; 5,861,760; 5,859,544; 5,859,542; 5,850,365; 5,850,152; 5,850,151; 5,848,005; 5,847,617; 5,845,385; 5,844,854; RE35,977; 5,838,628; 5,838,584; 5,835,998; 5,834,849; 5,828,229; 5,825,197; 5,821,787: 5,821,773; 5,821,771; 5,815,726; 5,815,024; 5,815,003; 5,812,479; 5,812,450; 5,809,281; 5,809,034; 5,805,516; 5,802,540; 5,801,541; 5,796,267; 5,793,246; 5,790,469; 5,787,009; 5,771,264; 5,768,562; 5,768,372; 5,767,734; 5,764,583; 5,764,569; 5,764,080; 5,764,079; 5,761,099; 5,760,624; 5,757,207; 5,757,070; 5,744,991; 5,744,383; 5,740,110; 5,732,020; 5,729,495; 5,717,901; 5,705,939; 5,699,020; 5,699,312; 5,696,455; 5,693,540; 5,694,058; 5,691,653; 5,689,195; 5,668,771; 5,680,061; 5,672,985; 5,670,895; 5,659,717; 5,650,734; 5,649,163; 5,642,262; 5,642,082; 5,633,830; 5,631,576; 5,621,312; 5,614,840; 5,612,642; 5,608,337; 5,606,276; 5,606,266; 5,604,453; 5,598,109; 5,598,108; 5,592,106; 5,592,102; 5,590,305; 5,583,749; 5,581,501; 5,574,893; 5,572,717; 5,572,148; 5,572,067; 5,570,040; 5,567,177; 5,565,793; 5,563,592; 5,561,757; 5,557,217; 5,555,214; 5,550,842; 5,550,782; 5,548,552; 5,548,228; 5,543,732; 5,543,730; 5,541,530; 5,537,295; 5,537,057; 5,525,917; 5,525,827; 5,523,706; 5,523,247; 5,517,186; 5,498,975; 5,495,182; 5,493,526; 5,493,519; 5,490,266; 5,488,586; 5,487,143; 5,486,775; 5,485,103; 5,485,102; 5,483,178; 5,477,474; 5,473,266; 5,463,328, 5,444,394; 5,438,295; 5,436,575; 5,436,574; 5,434,514; 5,432,467; 5,414,312; 5,399,922; 5,384,499; 5,376,844; 5,371,422; 5,369,314; 5,359,243; 5,359,242; 5,353,248; 5,352,940; 5,309,046; 5,350,954; 5,349,255; 5,341,308; 5,341,048; 5,341,044; 5,329,487; 5,317,210; 5,315,172; 5,301,416; 5,294,975; 5,285,153; 5,280,203; 5,274,581; 5,272,368; 5,268,598; 5,266,037; 5,260,611; 5,260,610; 5,258,668; 5,247,478; 5,247,477; 5,243,233; 5,241,224; 5,237,219; 5,220,533; 5,220,214; 5,200,920; 5,187,392; 5,166,604; 5,162,680; 5,144,167; 5,138,576; 5,128,565; 5,121,006; 5,111,423; 5,097,208; 5,091,661; 5,066,873; 5,045,772; 4,969,121; 4,930,107; 4,930,098; 4,930,097; 4,912,342; 4,903,223; 4,899,070; 4,899,067; 4,871,930; 4,864,161; 4,831,573; 4,785,423; 4,774,421; 4,713,792; 4,677,318; 4,617,479; 4,609,986; 4,020,469; and certain foreign patents.

Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights.

Printed on Recycled Paper.

I.S. EN ISO 9001

Contents

Tutorial Overview

This tutorial introduces you to the basic features of the Quartus™ programmable logic development system. It shows you how to create and process your own logic designs quickly and easily.

The sessions in this tutorial show you how to create and process a finite impulse response (FIR) filter design, called **fir_filter**. You will create several design files using text and graphic design entry. This tutorial describes a "top-down" hierarchical design method, in which you create a top-level block diagram first, and then create the lower-level designs. You will then compile all project files, specify timing requirements, and view timing analysis results. You will also perform a timing simulation and analyze the results. Finally, you will download the design into an Altera® APEX™ device.

The **fir_filter** tutorial is designed to help you learn how to use the Quartus software quickly. The tutorial is modular, so you can complete the sessions at your own pace; you can work through one session at a time, or complete the whole tutorial in one sitting. The tutorial is divided into the following five sections:

- "Design Entry" on page 8 teaches you how to create a top-level Block Design File (**.bdf**) using the Block Editor. You will also create several lower-level Verilog Design Files (**.v**) using the Text Editor and the MegaWizard™ Plug-In Manager.

- "Compilation" on page 51 teaches you how to compile a design using Compiler settings to control compilation processing. You will also learn to view the floorplan that shows how the Compiler placed logic in the device, and how to make location assignments.

- "Timing Analysis" on page 72 teaches you how to analyze the timing performance of logic in a design, including how to perform multi-clock timing analysis.

- "Simulation" on page 92 teaches you how to create a Vector Waveform File (**.vwf**) that contains input vectors for timing simulation. You will also learn how to create Simulator settings to control simulation processing and perform a timing simulation.

- "Programming" on page 108 teaches you how to use the Quartus Programmer to configure an Altera device.

Tutorial Files

The Quartus installation process copies all tutorial files to your hard disk, and creates the following directories at the same level as the **quartus** directory:

Directory Name:	**Description:**
\qdesigns	Quartus work directory.
\qdesigns\tutorial	Contains all files for this tutorial. This directory includes a **readme.txt** file that describes any changes to the tutorial since this manual was printed.
\qdesigns\fir_filter	Directory in which you should create the **fir_filter** project. Use this directory to prevent accidental changes to the original design files in the **\qdesigns\tutorial** directory.

☞ On a UNIX workstation, the **qdesigns** directory is a subdirectory of the **/usr** directory.

Command Shortcuts

Many Quartus commands have a variety of shortcuts. However, this tutorial focuses on the primary commands, and does not cover all of the shortcuts. For information on available mouse, keyboard, toolbar, and tool palette shortcuts, refer to Quartus menus, the **Toolbars** command (Tools menu), and Quartus online Help.

Getting Help

Throughout the tutorial, you can follow the footprints (👣) for useful references to Quartus Help. Online Help provides the most up-to-date and complete information on all Quartus features. Two of the easiest ways to get online Help are by using the context-sensitive Help feature, and the search index. You can also access Help by using the contents list or full-text search feature.

Context-Sensitive Help

Context-sensitive Help gives you instant help when you need it. You can access context-sensitive Help in these ways:

Method:	**Description:**
Shift+F1 keys	Press Shift+F1, point to an item on the screen, a text file keyword, or a menu command, and then click the left mouse button to obtain help.
F1 key	When a menu command is highlighted, a dialog box is open, or a pop-up message box is displayed, press F1 to obtain help. You can also press F1 when any Quartus application window is displayed to obtain general information about the context-sensitive Help available for that application.
Help command	When you want help on a message displayed in the Messages window, select the message and choose **Help** (right button pop-up menu).

Help Search Index

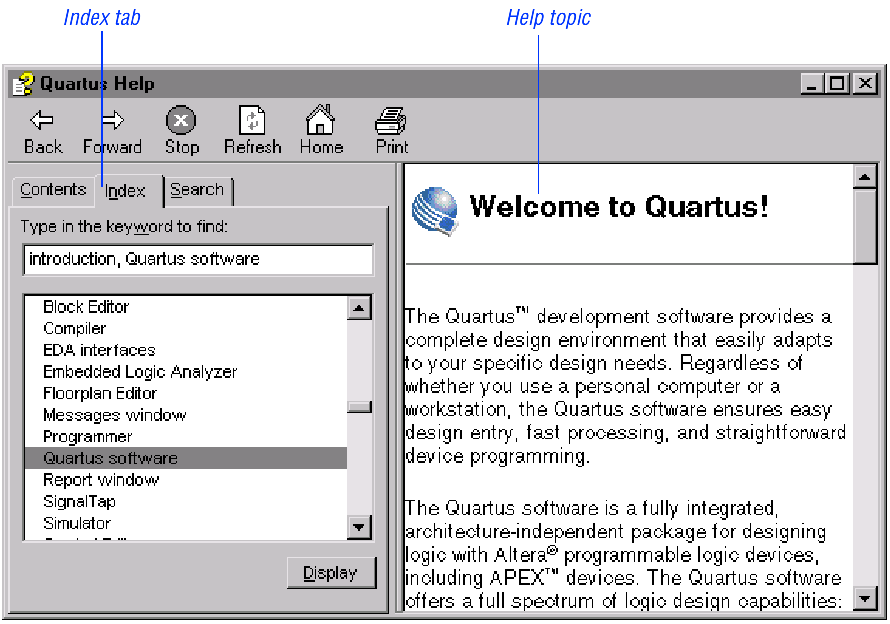
Quartus Help includes an extensive index to help you find information quickly. To search for a Help topic, follow these steps:

1. In the Quartus window, choose **Index** (Help menu).

   or

   If the Quartus Help window is already open, click the **Index** tab.

2. In the **Type in the keyword to find** box, type a keyword or phrase. The keyword list scrolls to display the keywords that match the text you type, as shown in the following illustration:



Index tab

Help topic

Type in the keyword to find: introduction, Quartus software

Block Editor
Compiler
EDA interfaces
Embedded Logic Analyzer
Floorplan Editor
Messages window
Programmer
Quartus software
Report window
SignalTap
Simulator

Welcome to Quartus!

The Quartus™ development software provides a complete design environment that easily adapts to your specific design needs. Regardless of whether you use a personal computer or a workstation, the Quartus software ensures easy design entry, fast processing, and straightforward device programming.

The Quartus software is a fully integrated, architecture-independent package for designing logic with Altera® programmable logic devices, including APEX™ devices. The Quartus software offers a full spectrum of logic design capabilities:

3. To display a topic, select a topic name and click **Display** or double-click a topic name.

Help Contents

Quartus Help includes a Contents list, to help you locate information by category. To use the Contents list, follow these steps:

1. In the Quartus window, choose **Contents** (Help menu).

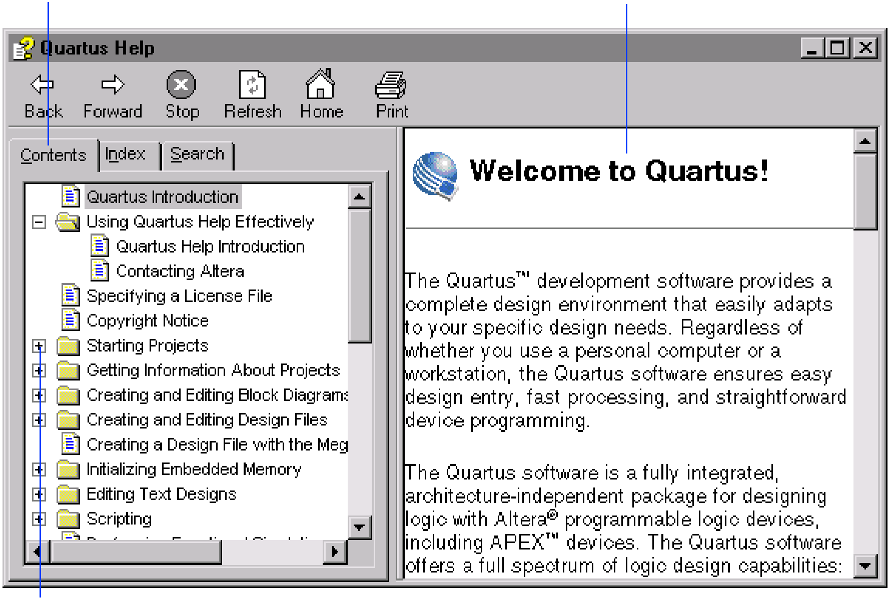
   or

   If the Quartus Help window is already open, click the **Contents** tab.

   See the following illustration:

Contents tab

Help topic


Quartus Help
Back
Forward
Stop
Refresh
Home
Print
Contents
Index
Search
Quartus Introduction
Using Quartus Help Effectively
Quartus Help Introduction
Contacting Altera
Specifying a License File
Copyright Notice
Starting Projects
Getting Information About Projects
Creating and Editing Block Diagrams
Creating and Editing Design Files
Creating a Design File with the Meg
Initializing Embedded Memory
Editing Text Designs
Scripting

Welcome to Quartus!

The Quartus™ development software provides a complete design environment that easily adapts to your specific design needs. Regardless of whether you use a personal computer or a workstation, the Quartus software ensures easy design entry, fast processing, and straightforward device programming.

The Quartus software is a fully integrated, architecture-independent package for designing logic with Altera® programmable logic devices, including APEX™ devices. The Quartus software offers a full spectrum of logic design capabilities:

Clicking the + icon expands the folder so that you can see its contents.

2. Click the + icon to expand the folder you want to open.

3. To display a topic, click the topic name.

Help Full-Text Search

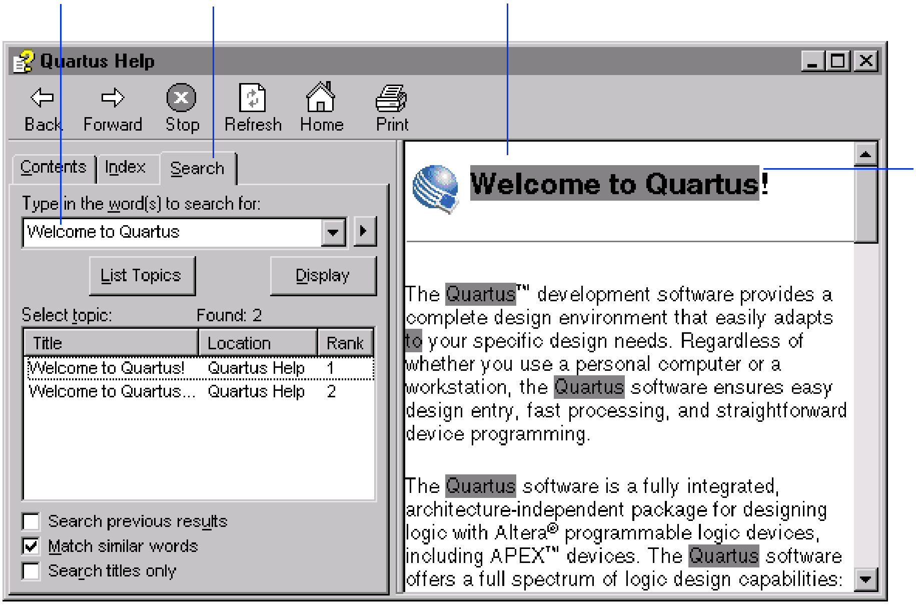
Quartus Help includes a full-text search to help you find the Help topics that contain the text you specify. To use the full-text search, follow these steps:

1. In the Quartus window, choose **Search** (Help menu).

   or

   If the Quartus Help window is already open, click the **Search** tab.

   See the following illustration:



Type in the word(s) to search for: Welcome to Quartus

Select topic: Found: 2

Title	Location	Rank
Welcome to Quartus!	Quartus Help	1
Welcome to Quartus...	Quartus Help	2

□ Search previous results
☑ Match similar words
□ Search titles only

Welcome to Quartus!

The Quartus™ development software provides a complete design environment that easily adapts to your specific design needs. Regardless of whether you use a personal computer or a workstation, the Quartus software ensures easy design entry, fast processing, and straightforward device programming.

The Quartus software is a fully integrated, architecture-independent package for designing logic with Altera® programmable logic devices, including APEX™ devices. The Quartus software offers a full spectrum of logic design capabilities:

Search words

Search tab

Help topic

Words matched in the search are highlighted in the Help topic.

2. In the **Type in the word(s) to search for** box, type the word(s) that you want to search for in Help.

3. To broaden or narrow a search, turn on one or more of the following options:

   – To limit a search to Help topics found in a previous search, turn on **Search previous results**.
   – To search for similar words, turn on **Match similar words**.

- To limit the search to Help topic titles only, turn on **Search titles only**.

4. To begin a search, click **List Topics**. The titles of the Help topics that match the search criteria appear in the **Select topic** list.

5. To display a topic, select a topic name and click **Display** or double-click a topic name.

Design Entry

In the Quartus software, a “project” consists of the complete set of design files, assignment files, simulation files, system settings, and hierarchy information for a design. The project you create in this tutorial demonstrates top-down design methodology. The following five tutorial sessions guide you through the steps needed to create the **fir_filter** tutorial project, and then explain how to create a top-level Block Design File (**.bdf**) that contains blocks representing the lower-level design files. Next, you create the lower-level Verilog Design Files (**.v**) that these blocks represent. You complete the design entry sessions by creating a custom megafunction variation that is instantiated by one of the lower-level design files.

Session 1: Start the Quartus Software

In this session, you will start the Quartus software to begin creating your project.

☞ This tutorial assumes that the Quartus working directory, which has the default name **qdesigns**, appears on the **d:** drive on your computer. If you installed the Quartus working directory in a different drive and/or directory, substitute the appropriate drive and/or directory name.

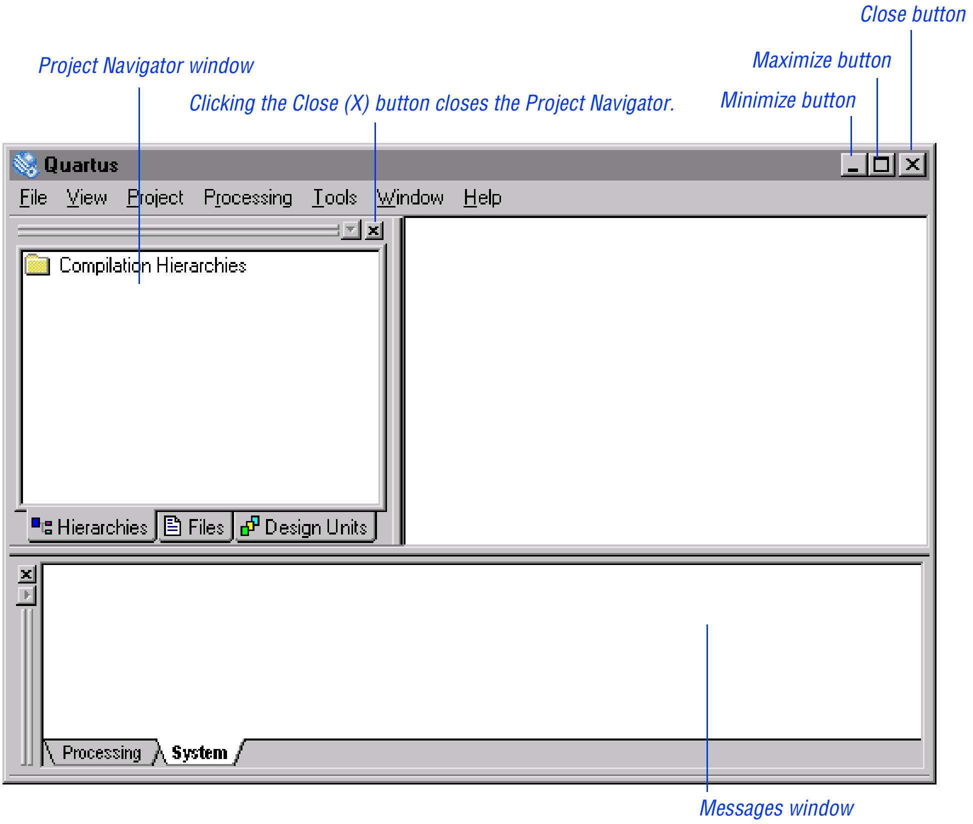
To start the Quartus software, follow these steps:

1. Choose **Programs** > **Altera** > **Quartus 1999.10** (Windows Start menu).

   or

   Type `quartus` ↵ at the command prompt. The Quartus window opens.

   See the following illustration:

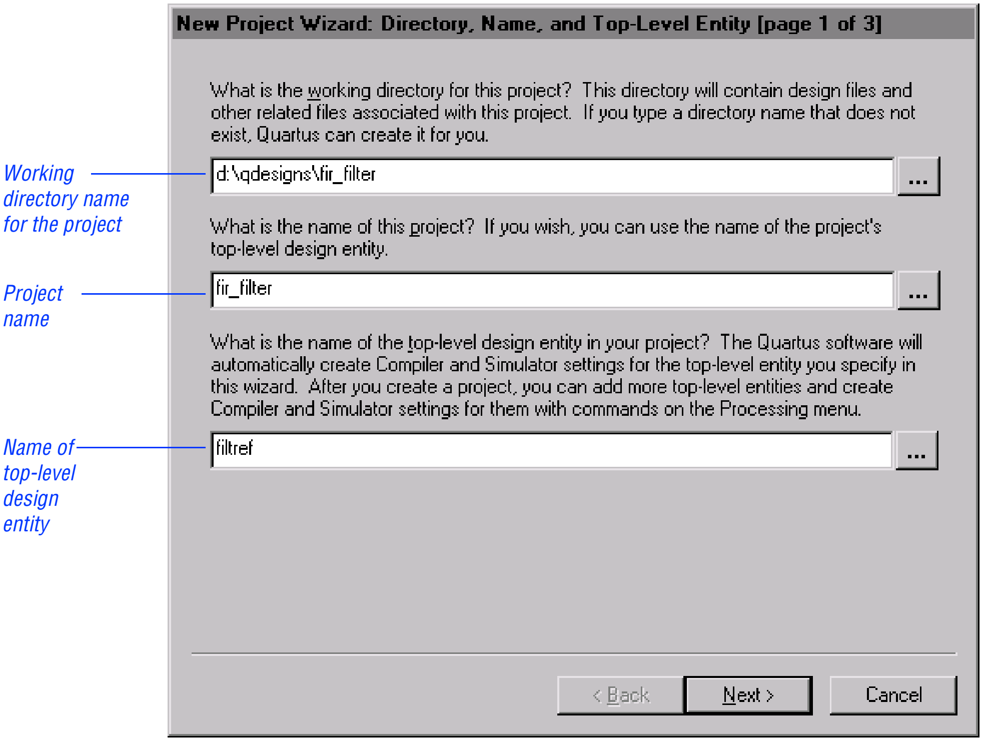

Close button
Maximize button
Project Navigator window
Clicking the Close (X) button closes the Project Navigator.
Minimize button
Quartus
File View Project Processing Tools Window Help
Compilation Hierarchies
Hierarchies Files Design Units
Processing System
Messages window

2. Maximize the Quartus window by clicking the **Maximize** button, as shown in the previous illustration.

Session 2: Create a Project

The Quartus software offers a **New Project** wizard to help you create a new project. To create a project using the **New Project** wizard, follow these steps:

1. Choose **New Project Wizard** (File menu). The **New Project** wizard appears. The first time you open the **New Project** wizard, it may display the Introduction page; you can click **Next** to proceed to the first page of the wizard. The following illustration shows the first page of the **New Project** wizard:



New Project Wizard: Directory, Name, and Top-Level Entity [page 1 of 3]

What is the working directory for this project? This directory will contain design files and other related files associated with this project. If you type a directory name that does not exist, Quartus can create it for you.

d:\qdesigns\fir_filter

What is the name of this project? If you wish, you can use the name of the project's top-level design entity.

fir_filter

What is the name of the top-level design entity in your project? The Quartus software will automatically create Compiler and Simulator settings for the top-level entity you specify in this wizard. After you create a project, you can add more top-level entities and create Compiler and Simulator settings for them with commands on the Processing menu.

filtref

Working directory name for the project

Project name

Name of top-level design entity

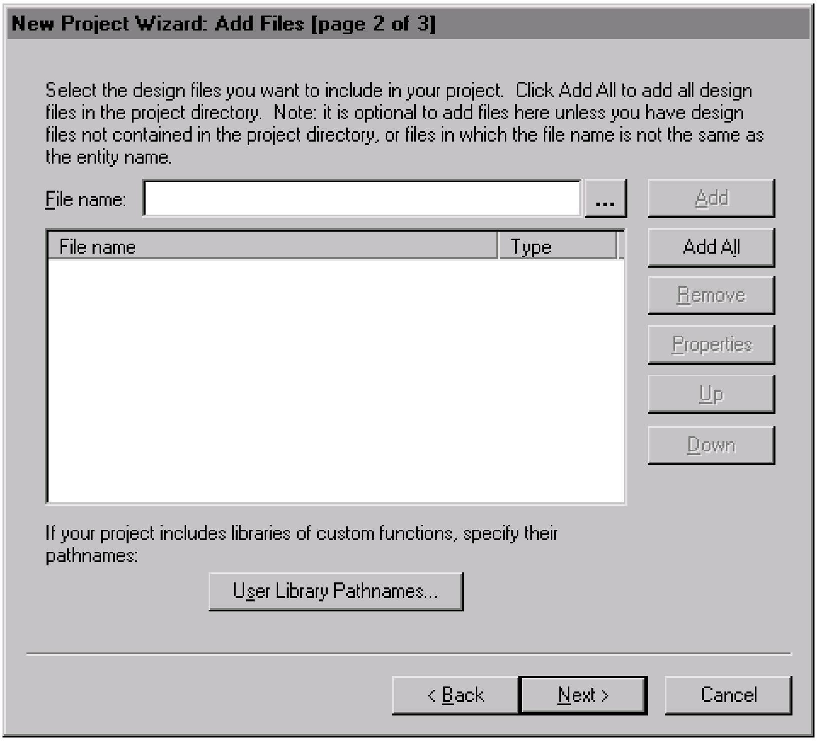
2. Type the directory name in the working directory box, or select the directory with **Browse** (**...**). For this example, type `d:\qdesigns\fir_filter` or browse to select it.

3. Type a name for the project in the project name box. For this example, type `fir_filter`.

4. Type `filtref` as the name of the top-level design entity of the project in the top-level design entity box.

   By default, the project name you enter appears as the name of the top-level design entity. However, you can use a different top-level design entity name, as this step directs you to do.

5. Click **Next**. The second page of the **New Project** wizard appears, as shown in the following illustration:



New Project Wizard: Add Files [page 2 of 3]

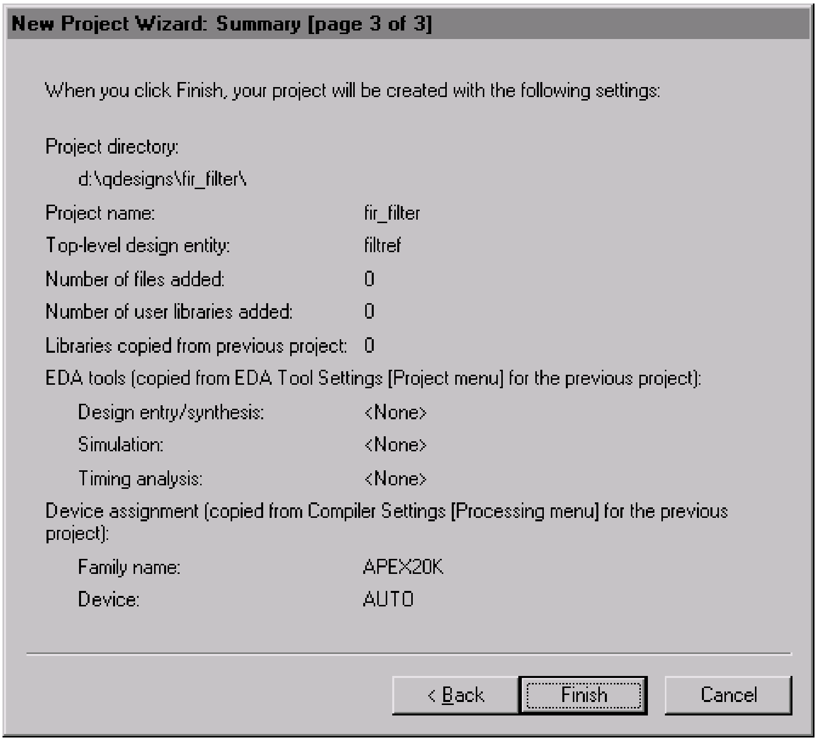
Select the design files you want to include in your project. Click Add All to add all design files in the project directory. Note: it is optional to add files here unless you have design files not contained in the project directory, or files in which the file name is not the same as the entity name.

If your project includes libraries of custom functions, specify their pathnames:

6. To add all of the design files in the project directory to the project, click **Add All**. Because **fir_filter** is a new project, the list is empty.

☞ If you are creating a project for which design files or other source files already exist, you can also use **Browse (...)** to select the files, and then click **Add** to add them to the project.

7. Click **Next**. The Summary page appears. The Summary page displays the project settings you specified in the wizard, such as the working directory, the name of the project, the top-level design entity, and the number of files in the project, as well as information about user libraries, EDA tools, and the device assignment. See the following illustration:



New Project Wizard: Summary [page 3 of 3]

When you click Finish, your project will be created with the following settings:

Project directory:
d:\qdesigns\fir_filter\

Project name:	fir_filter
Top-level design entity:	filtref
Number of files added:	0
Number of user libraries added:	0
Libraries copied from previous project:	0

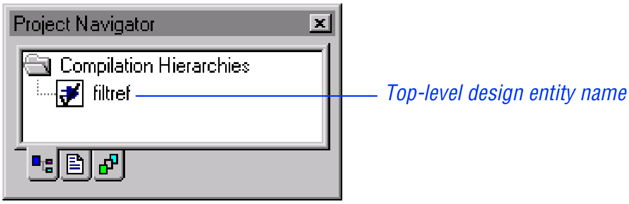
EDA tools (copied from EDA Tool Settings [Project menu] for the previous project):

Design entry/synthesis:	<None>
Simulation:	<None>
Timing analysis:	<None>

Device assignment (copied from Compiler Settings [Processing menu] for the previous project):

Family name:	APEX20K
Device:	AUTO

8. Click **Finish**. The project is now created. The top-level design entity name appears in the **Hierarchies** tab of the Project Navigator window. See the following illustration:



Top-level design entity name

Session 3: Create a Block Diagram

This session describes how to create a block diagram as the top-level design entity in the project. You will start by creating the top-level design entity, **filtref.bdf**.

If you are already familiar with design entry using the Quartus Block Editor, you can significantly reduce the time required to complete this tutorial by copying the following Altera-provided files from the **\qdesigns\tutorial** subdirectory into the **\qdesigns\fir_filter** subdirectory:

File Name:	**Description:**
filtref.bdf	Top-level Block Design File.
mult.v	Custom Verilog HDL variation of the `lpm_mult` megafunction created with the MegaWizard Plug-In Manager. This file is represented in the **filtref.bdf** schematic as `mult`.

Altera recommends that you copy tutorial files by opening them in the Quartus software with the **Open** command (File menu), choosing **Save As** (File menu), turning on **Add file to current project**, and then saving the file to the **\qdesigns\fir_filter** subdirectory. Once you have copied the Altera-provided files, you can skip to "Session 4: Create Verilog Design Files" on page 43.

This session includes the following steps:

1. Create a new Block Design File (**.bdf**).
2. Create a block.
3. Enter primitive and megafunction symbols.
4. Change the Block Editor display options.
5. Enter input and output pin symbols.
6. Name the pins.
7. Connect symbols and blocks.
8. Map signals between blocks.

1. Create a New Block Design File

In this step you will create a new Block Design File (**.bdf**) called **filtref.bdf**. This file is the top-level design entity of the **fir_filter** project.

To create a new BDF, follow these steps:

1. Choose **New** (File menu). The **Design Files** tab appears automatically.

2. In the **Design Files** tab, select **Block Diagram/Schematic File**.

3. Click **OK**. A new Block Editor window opens.

4. Choose **Save As** (File menu).

5. Select the folder where you want to save the BDF. The **Save As** dialog box should automatically display the project directory name, **d:\qdesigns\fir_filter**, as the directory for saving the file.

6. In the **File name** box, type `filtref` as the name, if necessary.

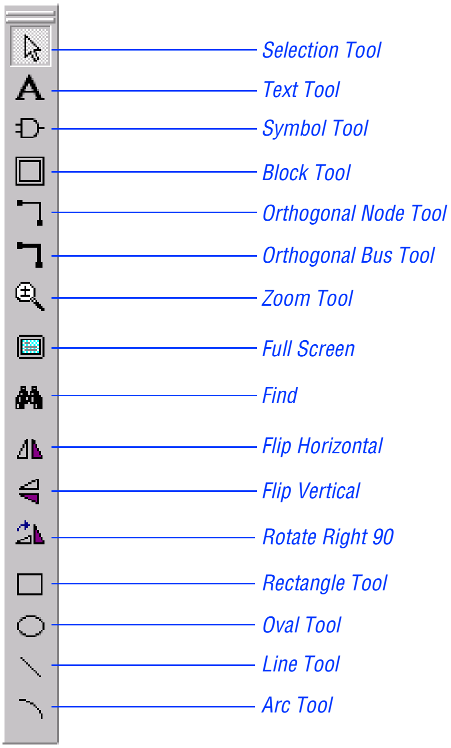
7. If necessary, turn on **Add file to current project**.

8. Click **Save**. The file is saved and added to the project.

2. Create a Block

Follow these steps to create a new block in the **filtref.bdf** file:

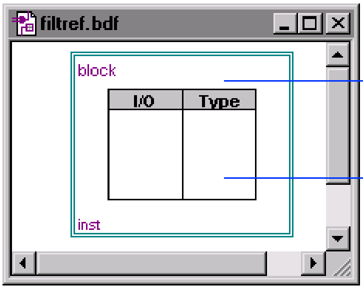
1. In the Block Editor window, click the **Block Tool** button on the Block Editor toolbar. The Block Editor window has the following default toolbar buttons:



2. Click in a blank space in the Block Editor and drag the Block Tool pointer until the block is the size you want.

☞ You can use the **Undo** and **Redo** commands from the Edit menu to make corrections, if necessary.

See the following illustration:



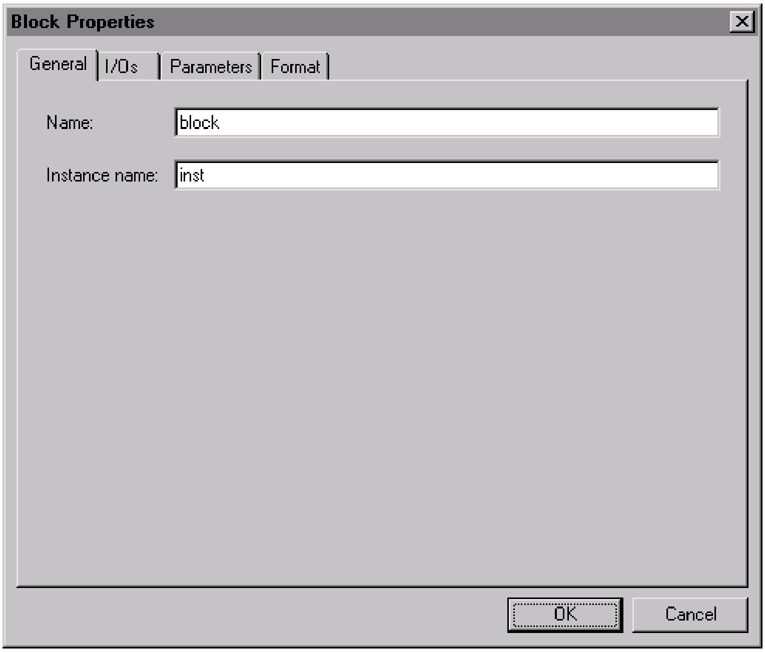
Block symbol

I/O ports table for the block (empty because ports have not been specified yet).

3. Click the **Selection Tool** (arrow) button on the Block Editor toolbar.

4. With the Selection Tool, double-click the new block. The **Block Properties** dialog box appears.

5. Click the **General** tab. See the following illustration:



Name: block

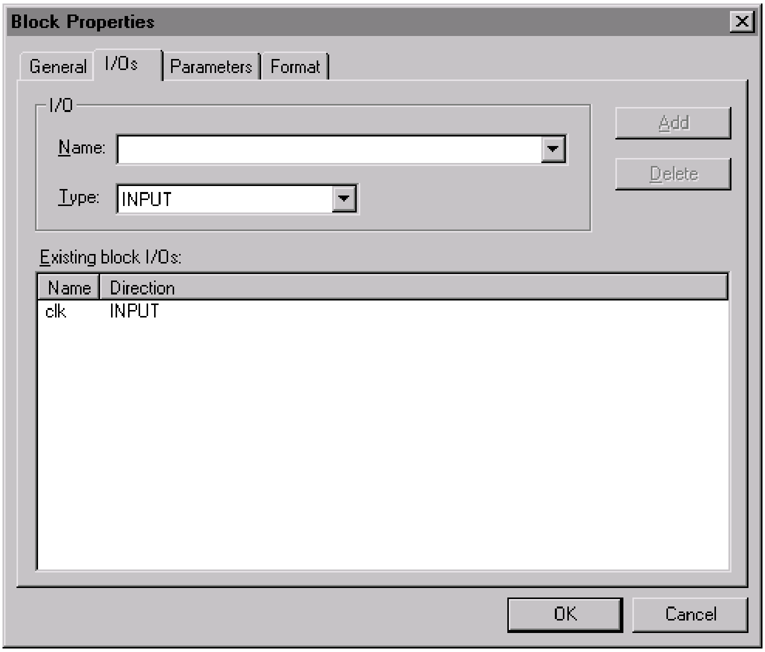
Instance name: inst

6. In the **Name** box, type `taps` as the block name, and keep the default instance name, `inst`.

7. Click the **I/Os** tab.

8. Under **I/O**, type `clk` in the **Name** box as the first port name, and in the **Type** list, select **INPUT**.

9. Click **Add**. The `clk` port name appears in the **Existing block I/Os** list, as shown in the following illustration:



Type: INPUT

Name	Direction
clk	INPUT

10. Repeat steps 8 and 9 to enter each of the port names shown in the following table:

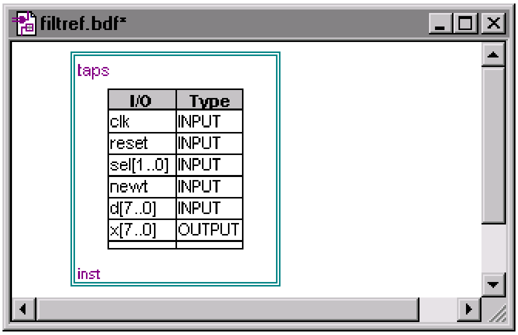
You can type in all of the input port names at once, separating each name by a comma, and then click **Add**.

Name:	**Type:**
`clk` (already entered)	INPUT
`reset`	INPUT
`sel[1..0]`	INPUT
`newt`	INPUT
`d[7..0]`	INPUT
`x[7..0]`	OUTPUT

11. Click **OK**. You have now specified the input and output ports of the taps block. In session 4, you will learn how to create the design file that the taps block represents.

12. Select the taps block.

13. Choose **AutoFit** (right button pop-up menu). This command resizes the border of the taps block to fit proportionally around the I/O ports table, and ensures that all data about the block is visible. See the following illustration:



I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

14. Repeat steps 1 through 13 to create three more blocks—state_m, hvalues, and acc—with the ports listed in the following tables.

    Specify these ports for the state_m block:

Name:	**Type:**
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPUT
next	OUTPUT
first	OUTPUT

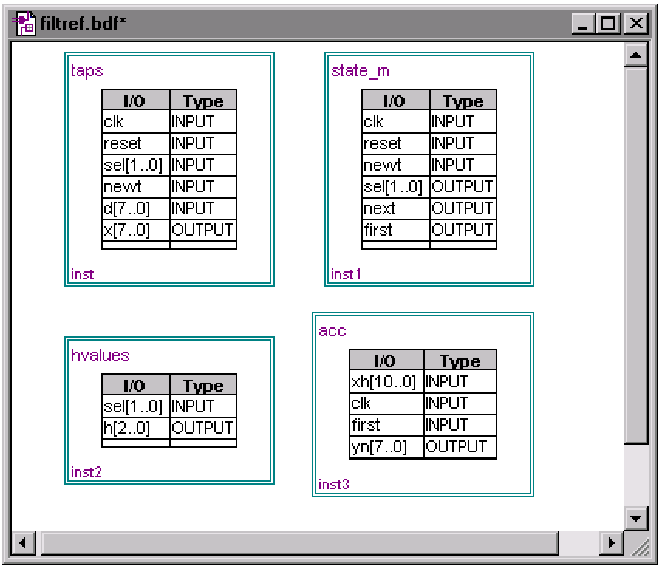
Specify these ports for the `hvalues` block:

Name:	Type:
`sel[1..0]`	INPUT
`h[2..0]`	OUTPUT

Specify these ports for the `acc` block:

Name:	Type:
`xh[10..0]`	INPUT
`clk`	INPUT
`first`	INPUT
`yn[7..0]`	OUTPUT

15. To move any of the blocks to a new location, click the **Selection Tool** button on the Block Editor toolbar or press Esc to activate the Selection Tool, and then drag the block to a new location. When you have finished, the four blocks should appear roughly as shown in the following illustration:



taps

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

inst

state_m

I/O	Type
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPUT
next	OUTPUT
first	OUTPUT

inst1

hvalues

I/O	Type
sel[1..0]	INPUT
h[2..0]	OUTPUT

inst2

acc

I/O	Type
xh[10..0]	INPUT
clk	INPUT
first	INPUT
yn[7..0]	OUTPUT

inst3

16. Choose **Save** (File menu).

3. Enter Primitive & Megafunction Symbols

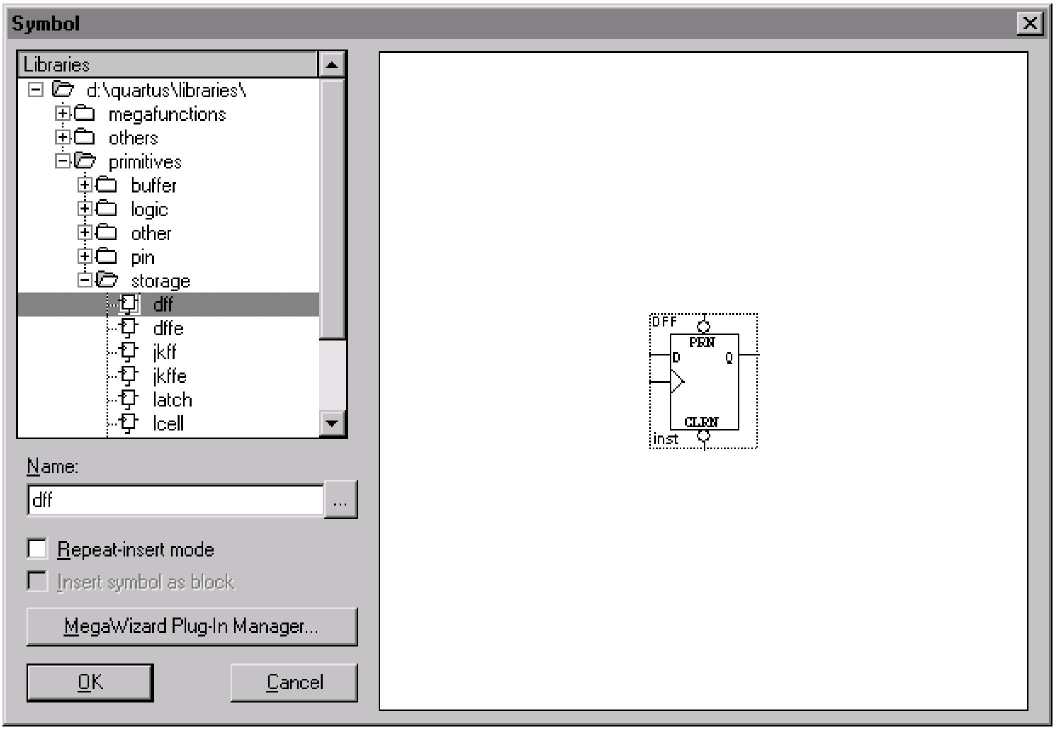
A Block Design File (**.bdf**) can contain both blocks, like those you have already created in this tutorial, and "ordinary" schematic symbols. The Quartus software provides symbols for a variety of logic functions—including primitives, Library of Parameterized Modules (LPM) functions, and other megafunctions—that you can use in the Block Editor.

Follow these steps to enter a `DFF` (D flipflop) symbol in the **filtref.bdf** file:

1. Double-click in empty space in the Block Editor window.

2. In the **Symbol** dialog box, in the **Libraries** list, click the + icon to expand the **d:\quartus\libraries** folder. Similarly, expand the **primitives** folder, and then expand the **storage** folder.

3. In the storage **folder**, select the **dff** primitive. A preview of the new symbol appears in the **Symbol** dialog box, as shown in the following illustration:



Name: dff

□ Repeat-insert mode

□ Insert symbol as block

☞ As an alternative to steps 2 and 3, you can simply type `dff` in the **Name** box.

4. Click **OK**. An outline of the `DFF` symbol is now attached to the pointer.

5. Click the pointer at the desired location in the Block Editor window to insert the `DFF` symbol into the design file.

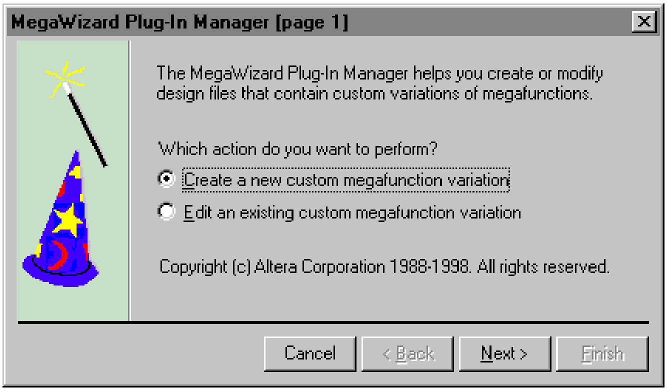
6. Repeat steps 1 through 5 to enter a `DFFE` symbol in the **filtref.bdf** file.

The next symbol you need to enter is a multiplier that is a variation of the `lpm_mult` megafunction. You can use the MegaWizard Plug-In Manager to enter this symbol. The MegaWizard Plug-In Manager allows you to create (or modify) design files that contain custom variations of megafunctions. These custom megafunction variations are based on Altera-provided megafunctions, including Library of Parameterized Modules (LPM) functions. The MegaWizard Plug-In Manager runs a wizard that helps you specify options for customization easily. The wizard prompts you about the values you want to set for parameters and which optional ports you want to use. Once the wizard generates the multiplier variation, you can instantiate it in the design file.

To enter a `mult` symbol generated by the MegaWizard Plug-In Manager, follow these steps:

1. Double-click an empty space in the Block Editor window.

2. In the **Symbol** dialog box, click **MegaWizard Plug-In Manager**. The first page of the MegaWizard Plug-In Manager is displayed, as shown in the following illustration:



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

Copyright (c) Altera Corporation 1988-1998. All rights reserved.

3. When you are asked **Which action do you want to perform?**, select **Create a new custom megafunction variation** and click **Next**.

4. In the **Available Megafunctions** list, click the + icon to expand the **arithmetic** folder, then select **LPM_MULT**.

5. When you are asked **Which type of output file do you want to create?**, select **Verilog HDL**.

6. When you are asked **What name do you want for the output file?**, specify **d:\qdesigns\fir_filter\mult.v** and click **Next**.

7. When you are asked **How wide should the 'dataa' input bus be?**, select **8**.

8. When you are asked **How wide should the 'datab' input bus be?**, select **3**.

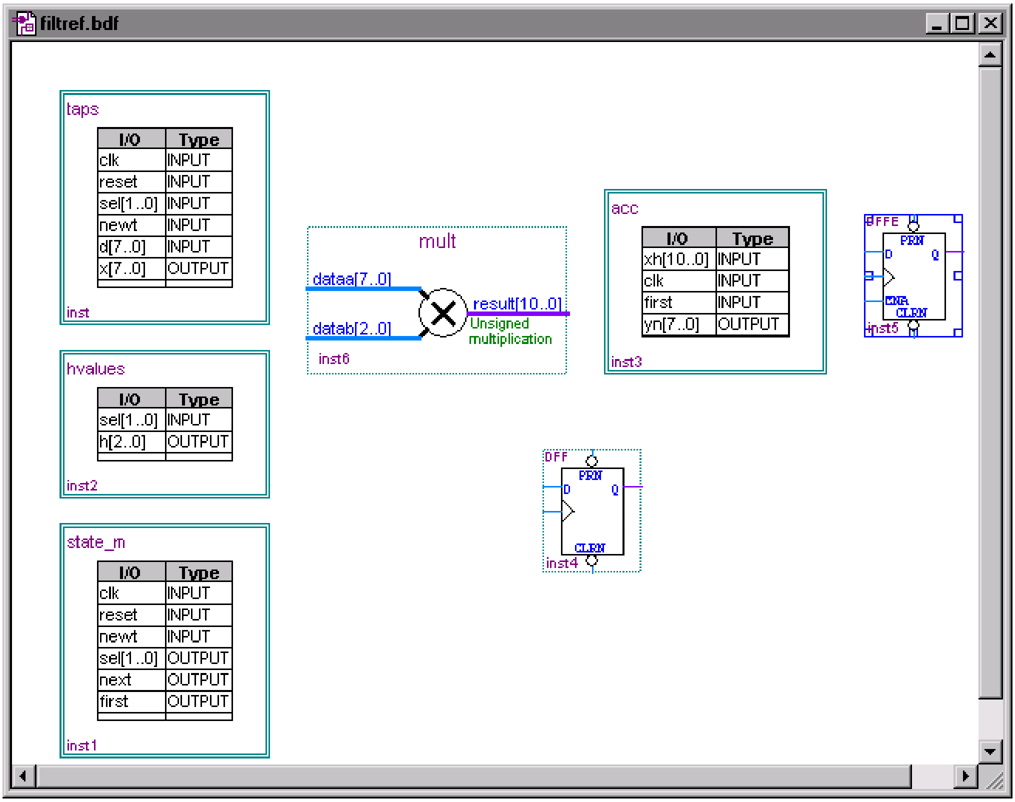
9. To accept the defaults for the remaining questions and generate the symbol, click **Finish**. A preview of the new symbol appears in the **Symbol** dialog box.

10. Click **OK**. An outline of the `mult` symbol is attached to the pointer.

11. To place the symbol, click an empty space in the Block Editor window.

12. Choose **Save** (File menu).

The following illustration shows the correct arrangement of the four block symbols as well as the `DFF`, `DFFE`, and `mult` symbols:



taps

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

inst

hvalues

I/O	Type
sel[1..0]	INPUT
h[2..0]	OUTPUT

inst2

state_m

I/O	Type
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPUT
next	OUTPUT
first	OUTPUT

inst1

acc

I/O	Type
xh[10..0]	INPUT
clk	INPUT
first	INPUT
yn[7..0]	OUTPUT

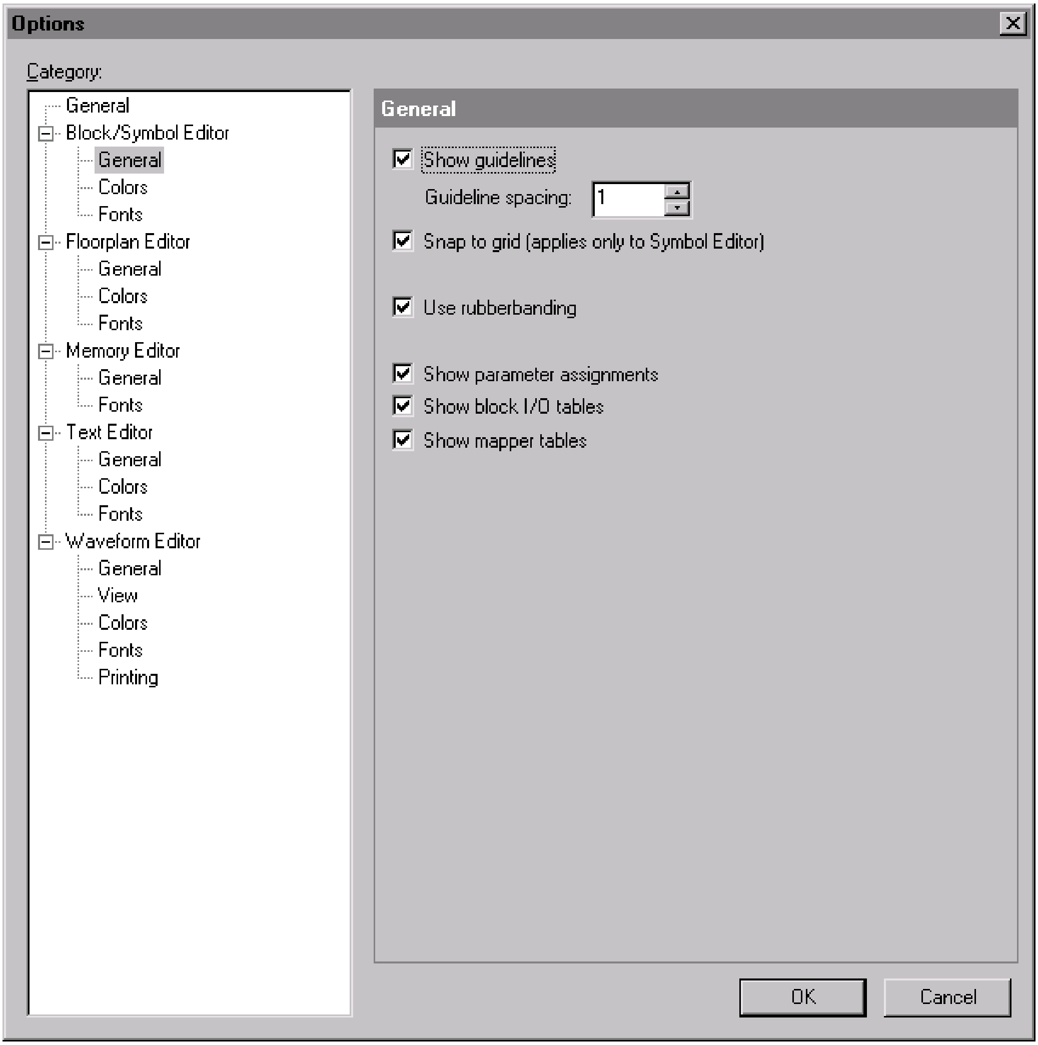
inst3

4. Change the Block Editor Display Options

You can change the Block Editor display options, as needed. To change the Block Editor display options, follow these steps:

1. Choose **Options** (Tools menu).

2. In the **Category** list, under **Block/Symbol Editor**, select **General**. See the following illustration:



☑ Show guidelines

Guideline spacing: 1

☑ Snap to grid (applies only to Symbol Editor)

☑ Use rubberbanding

☑ Show parameter assignments

☑ Show block I/O tables

☑ Show mapper tables

3. In the **General** tab, turn appropriate settings on or off, according to your preferences.

4. To modify the colors and fonts used in the Block Editor window, in the **Category** list, select **Colors** or **Fonts**.

5. Click **OK**.

You can also view larger or smaller portions of the file with the Zoom Tool, which is available by clicking the **Zoom Tool** button on the Block Editor toolbar, and with the **Zoom**, **Zoom In**, **Zoom Out**, and **Fit in Window** commands from the View menu.

5. Enter Input & Output Pin Symbols

To enter input and output pins, follow these steps:

1. Click the **Symbol Tool** button on the Block Editor toolbar. The same **Symbol** dialog box that you used to enter the `DFF` and `DFFE` symbols appears. Note, however, that using the toolbar button opens this dialog box with the **Repeat-insert mode** option turned on.

   ☞ When **Repeat-insert mode** is turned on, an outline of the selected symbol remains attached to the pointer, regardless of how many times you click the mouse pointer, allowing you to place multiple copies of the symbol easily. Whenever you want to stop placing copies of a symbol, you can press Esc or choose **Cancel** (right button pop-up menu).

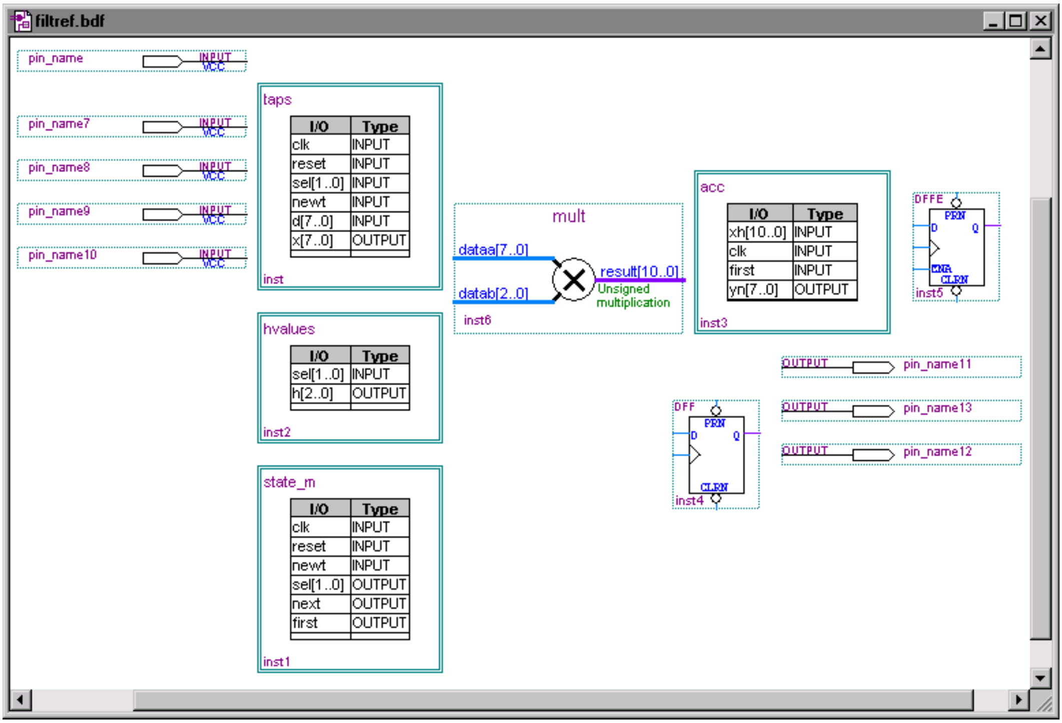
2. In the **Symbol** dialog box, in the **Libraries** list, click the + icon to expand the **d:\quartus\libraries** folder, expand the **primitives** folder, and then expand the **pin** folder.

3. In the **pin** folder, select the **input** primitive.

4. Click **OK**.

5. Click in empty space five times to insert a total of five `INPUT` symbols on the left-hand side of the file. Symbols are automatically named as `pin_name`<*number*> in sequence. Press Esc.

6. Repeat steps 1 to 5 to insert and position a total of three `OUTPUT` symbols on the right-hand side of the file. See the following illustration:



taps

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

inst

hvalues

I/O	Type
sel[1..0]	INPUT
h[2..0]	OUTPUT

inst2

state_m

I/O	Type
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPUT
next	OUTPUT
first	OUTPUT

inst1

acc

I/O	Type
xh[10..0]	INPUT
clk	INPUT
first	INPUT
yn[7..0]	OUTPUT

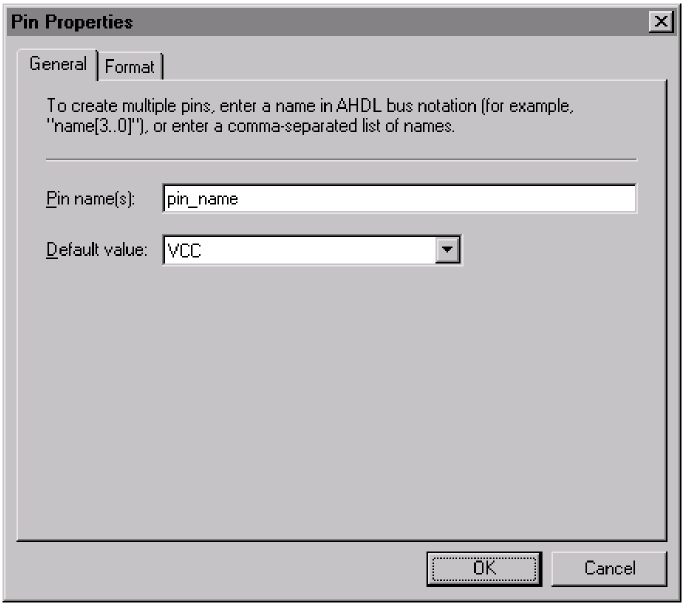
inst3

7. Choose **Save** (File menu).

6. Name the Pins

You will now name the input and output pins. To name a pin, follow these steps:

1. With the Selection Tool, double-click the first input pin symbol you entered. The **General** tab of the **Pin Properties** dialog box appears automatically. See the following illustration:



Pin Properties

General | Format

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): pin_name

Default value: VCC

2. In the **Pin name(s)** box, type `clkx2` to replace the default name of the first pin, that is, to replace `pin_name`.

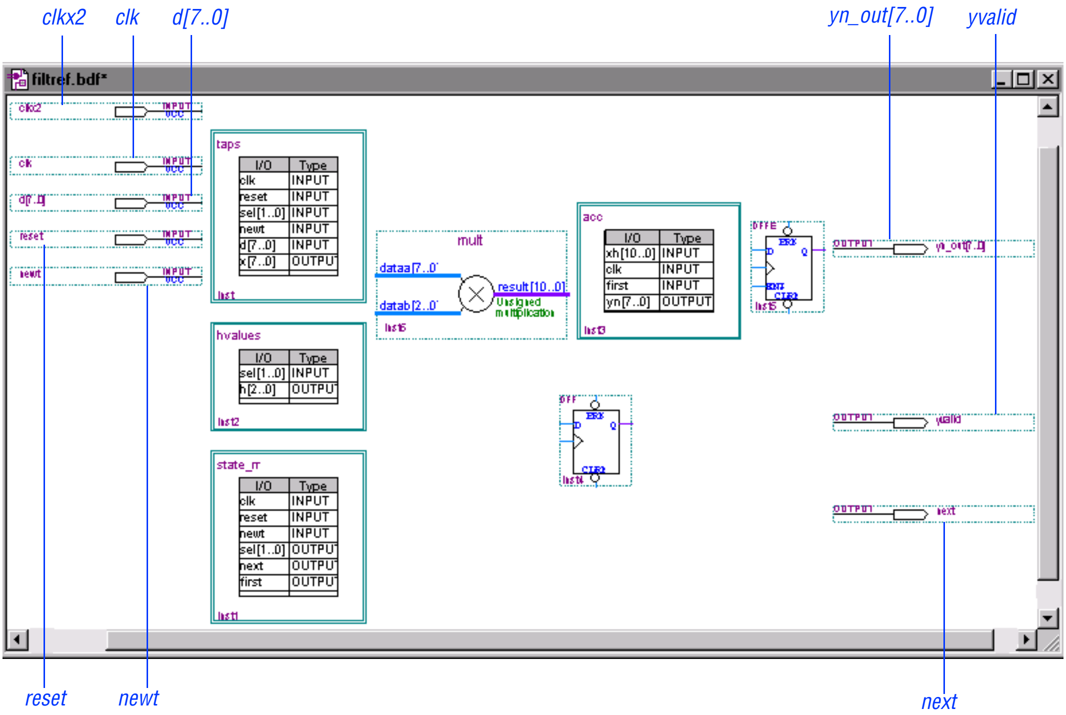
3. Click **OK**.

4. Repeat steps 1 to 3 to rename each of the pins with the following names:

Pin Type:	**Rename As:**	**Description:**
`INPUT`	`clkx2` (already entered)	Derived clock for the FIR filter.
`INPUT`	`clk`	Base clock for FIR filter.
`INPUT`	`d[7..0]`	Data input to the FIR filter.
`INPUT`	`reset`	Reset signal for the FIR filter.
`INPUT`	`newt`	Input signal that loads the data input `d[7..0]` into the `taps` function.

Pin Type:	Rename As:	Description:
`OUTPUT`	`yn_out[7..0]`	The FIR filter output data.
`OUTPUT`	`yvalid`	Indicates that the `yn[7..0]` filter output of the `acc` function is valid.
`OUTPUT`	`next`	Indicates that the FIR filter is ready for the next 8-bit data input.

5. Move the `INPUT` and `OUTPUT` pin symbols so they line up with the appropriate symbols or blocks, as shown in the following illustration:



taps

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPU

hvalues

I/O	Type
sel[1..0]	INPUT
h[2..0]	OUTPU

state_m

I/O	Type
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPU
next	OUTPU
first	OUTPU

acc

I/O	Type
xh[10..0]	INPUT
clk	INPUT
first	INPUT
yn[7..0]	OUTPUT

6. Choose **Save** (File menu).

7. Connect Symbols & Blocks

You can use the “smart” Selection Tool to draw most of the lines you need to connect symbols and blocks in a BDF. The lines that you draw to connect symbols include nodes, buses, and “conduits.” Conduits are bus lines that represent one or more signals traveling to or from a block.

The “smart” Selection Tool pointer automatically turns into an appropriate line-drawing pointer when it is over a symbol pinstub or a block border. For example, it turns into the Orthogonal Node Tool pointer when you point it at a symbol pinstub, and turns into the Orthogonal Bus Tool pointer when you point it at a block border. You can also draw lines of a particular type with the Orthogonal Node Tool and Orthogonal Bus Tool.

The Quartus software makes connections automatically between blocks that have been connected together, or between the signals on a bus and the names of I/Os in blocks, so the bus acts as a conduit for any number of signals. Go to “8. Map Signals between Blocks” on page 35 for more information about conduits and about mapping signals between blocks.

To draw a bus or conduit line, follow these steps:

1. Click the **Orthogonal Bus Tool** button on the Block Editor toolbar.

2. Click the pinstub of the `clk` input pin to define the start of the bus, and then drag the pointer to draw a line that connects to the border of the `taps` block. A “mapper” symbol appears automatically on the edge of the `taps` block where the bus connects to the block. A mapper allows you to map I/O port(s) in the block to signal(s) in the bus. Refer to “8. Map Signals between Blocks” on page 35 for more information about using block mappers.

3. Repeat steps 1 through 2 to make the additional connections between symbols and blocks shown in the following table. You can also refer to Figure 1 on page 34, or to the Altera-provided **filtref.bdf** file.

   ☞ To draw a line that connects a node or bus to an existing node or bus, you can choose the **Orthogonal Node Tool** or **Orthogonal Bus Tool** button on the Block Editor toolbar (see step 1 on page 15 for an illustration of the toolbar buttons). When you draw a line that connects a node or bus to an existing node or bus, a connection “dot” appears.

Draw Line From:	To:
INPUT pin clk	taps block (already entered)
Bus connecting INPUT pin clk to taps block	state_m block
INPUT pin d[7..0]	Bus connecting taps block to state_m block
INPUT pin reset	Bus connecting taps block to state_m block
INPUT pin newt	Bus connecting taps block to state_m block
state_m block	OUTPUT pin next
Q output of DFFE primitive	OUTPUT pin yn_out[7..0]

4. Click the **Selection Tool** button on the Block Editor toolbar.

5. Repeat step 2 to make the additional connections between symbols and blocks shown in the following table. You can also refer to Figure 1 on page 34, or to the Altera-provided **filtref.bdf** file.

Draw Line From:	To:
taps block	hvalues block
Bus connecting taps block to hvalues block	state_m block
Bus connecting INPUT pin clk to taps block	acc block
state_m block	acc block
taps block	dataa[7..0] input of mult symbol

Draw Line From:	**To:**
`hvalues` block	`datab[2..0]` input of `mult` symbol
`result[10..0]` output of `mult` symbol	`acc` block

6. Click the **Orthogonal Node Tool** button on the Block Editor toolbar.

7. Draw a line from the `D` input of the `DFF` symbol to the border of the `state_m` block.

8. Repeat step 7 to create the additional conduit connections between blocks and symbols, as shown in the following table. You can also refer to Figure 1 on page 34, or to the Altera-provided **filtref.bdf** file.

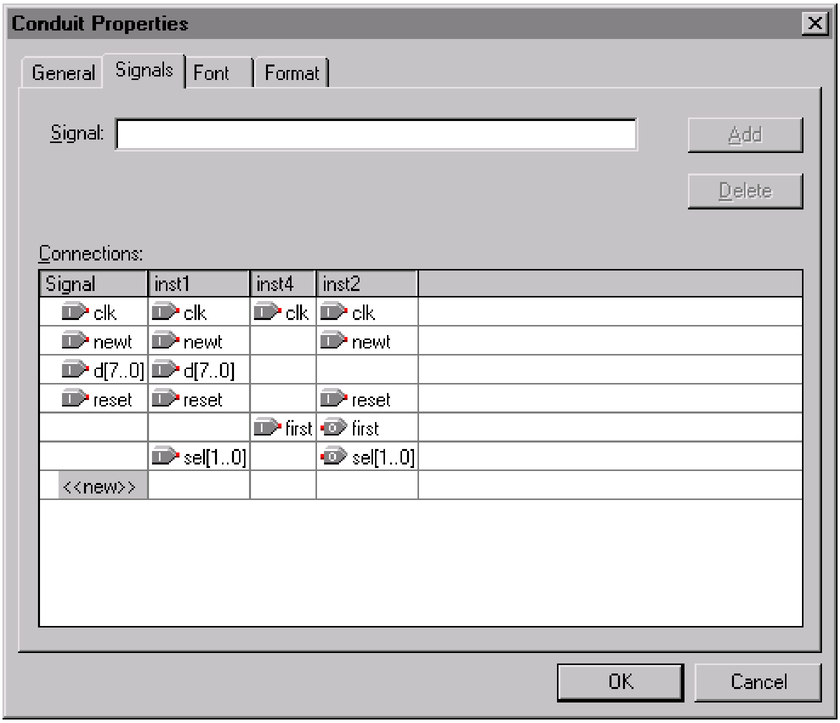
Draw Conduit From:	**To:**
`state_m` block	`D` input of the `DFF` symbol (already entered)
`acc` block	`D` input of the `DFFE` symbol

☞ The connection from the `state_m` block to the `D` input of the `DFF` symbol is considered to be a conduit, because it is connected to a block; however, because it contains a single signal, it is acceptable to draw this line with a thin orthogonal node line, rather than a thick bus line.

9. Select the bus (or conduit) that connects the `INPUT` pin `clk` to the `taps` block, and choose **Properties** (right button pop-up menu).

10. In the **Conduit Properties** dialog box, click the **Signals** tab.

The **Connections** list shows the signal connections for the selected conduit. See the following illustration:



Signal	inst1	inst4	inst2	
clk	clk	clk	clk	
newt	newt		newt	
d[7..0]	d[7..0]			
reset	reset		reset	
		first	first	
	sel[1..0]		sel[1..0]	
<<new>>				

By default, the output signals appear in colors that are different from the input signal colors.

11. Verify that the signals are correct, and then click **OK**.

To draw a node line, follow these steps:

1. Click the **Selection Tool** button on the Block Editor toolbar.

2. Draw a line from the pinstub of the `Q` output of the `DFF` symbol to the input pinstub of the `OUTPUT` pin `yvalid`.

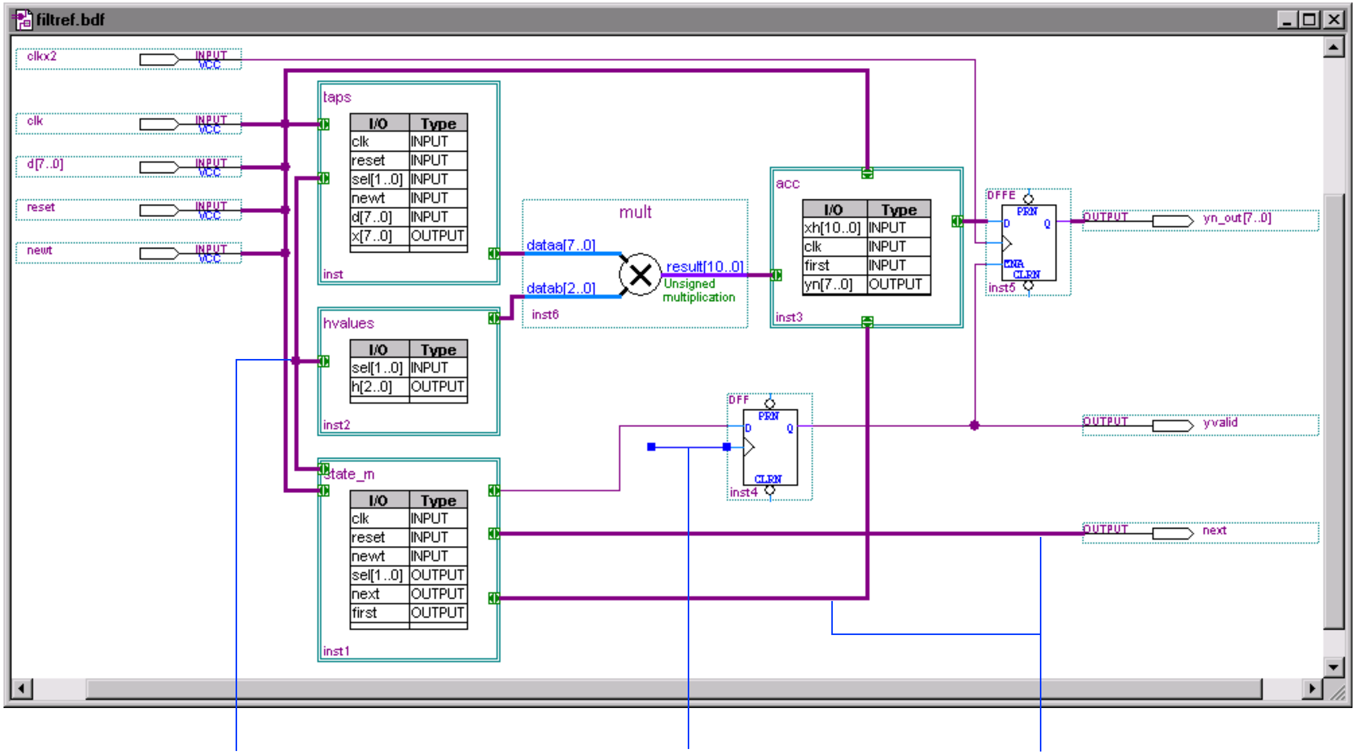
3. Repeat steps 1 through 2 to make the additional connections between pins and primitives, as shown in the following table:

Draw Line From:	**To:**
`Q` output of `DFF` primitive	`OUTPUT` pin `yvalid` (already connected)
`INPUT` pin `clkx2`	`clock` input of `DFFE` primitive
node connecting the `Q` output of the `DFF` primitive to `OUTPUT` pin `yvalid`	`enable` (`ENA`) input of the `DFFE` primitive

4. To create a node that can be connected by name, draw a line from the `clock` input to the `DFF` symbol into an empty space. In "8. Map Signals between Blocks" on page 35, you will learn how to complete the connection by name for this node.

5. Choose **Save** (File menu).

Figure 1. Node & Bus Connections for the filtref.bdf Block Design File



taps

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

hvalues

I/O	Type
sel[1..0]	INPUT
h[2..0]	OUTPUT

state_m

I/O	Type
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPUT
next	OUTPUT
first	OUTPUT

acc

I/O	Type
xh[10..0]	INPUT
clk	INPUT
first	INPUT
yn[7..0]	OUTPUT

Connection dot

Node to be connected later by name

Bus or “conduit”

8. Map Signals between Blocks

The Quartus software offers different ways to map signals between blocks and symbols:

Mapping Method: **Description:**

“Smart” mapping

If the I/O signal names in one block are the same as in another block, the common I/Os between the blocks are connected automatically. You do not need to label these buses or conduits. To prevent an automatic mapping, you can specify that a particular block’s I/O port should be mapped to “nothing.”

Assigning names to nodes or buses (including “connection by name”)

If the I/O names are different between the blocks or symbols you want to connect, you can assign a name to the conduit that matches the name of one of the block’s I/O ports to establish a mapping between that port and the bus or conduit.

Assigning a name is especially useful if you want to connect a conduit to another conduit, even though those conduits are not physically connected. To create a mapping between these conduits and their blocks, you must first make sure that each conduit is connected to a block on one end, but is physically unconnected to any other block or symbol. If you then assign matching names to both conduits (assuming that those names also match I/O ports in the connected blocks), you can establish a “connection by name” between the two conduits and their blocks. You can also use a similar method to connect two nodes by name.

Using “mappers” to specify mappings explicitly

If the I/O names are different in the blocks you want to connect, and the blocks are physically connected, you can specify the mappings explicitly. You can map the block I/O name to a signal on the bus and map that signal to the block I/O name you want to connect.

This section explains which signals are connected automatically through "smart" mapping, and shows how to map a total of six signals that are not mapped automatically. This section also shows how to assign appropriate names to two signals in order to establish connections, and how to specify explicit mappings for four other signals.

In the **filtref.bdf** design, the following connections make use of "smart" block mapping—*you do not need to name these connections*:

From:	**To:**
`INPUT` pin `clk`	Block I/Os named `clk` in blocks that are connected to the `clk` pin
`INPUT` pins `d[7..0]`	Block I/Os named `d[7..0]` in the `taps` block
`INPUT` pin `reset`	Block I/Os named `reset` in the `taps` and `state_m` blocks
`INPUT` pin `newt`	Block I/Os named `newt` in the `taps` and `state_m` blocks
Block I/O named `sel[1..0]` in the `taps` block	Block I/Os named `sel[1..0]` in the `hvalues` and `state_m` blocks
Block I/O named `first` in the `state_m` block	Block I/Os named `first` in the `acc` block
Block I/O named `next` in `state_m` block	`OUTPUT` pin `next`

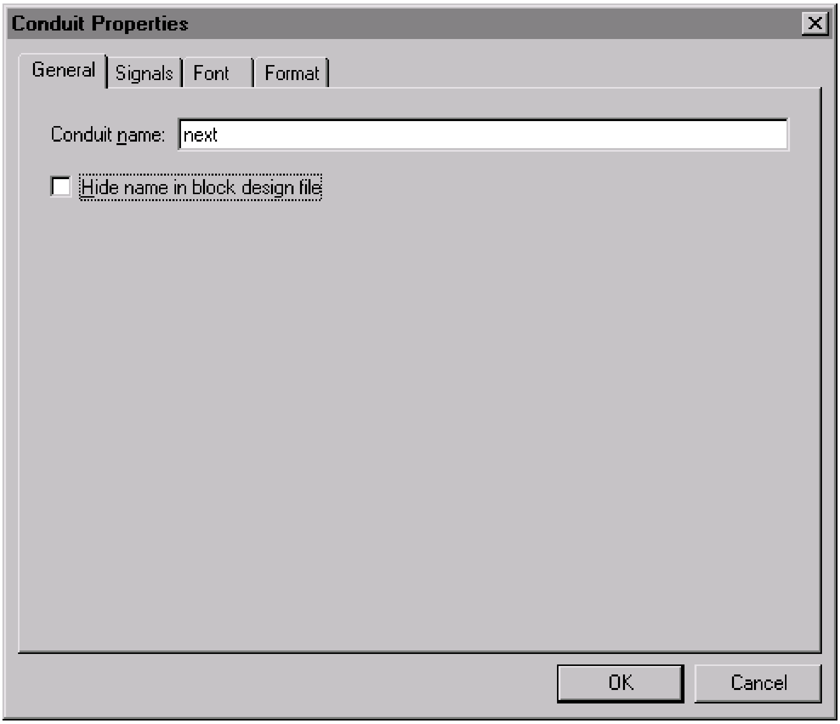
To assign appropriate names to some of the signals so they can be logically connected, follow these steps:

1. With the Selection Tool, select the conduit that connects the `state_m` block to the `D` input of the `DFF` primitive.

2. Choose **Properties** (right button pop-up menu).

3. In the **Conduit Properties** dialog box, if necessary, click the **General** tab.

4. In the **Conduit name** box, type `next` as the name of the conduit. See the following illustration:



Conduit name: next

□ Hide name in block design file

5. Click **OK**. The signal `next` is added automatically to the conduit, and the name appears above the conduit line. Adding this name creates a logical connection between the `state_m` block and the `D` input of the `DFF` primitive.

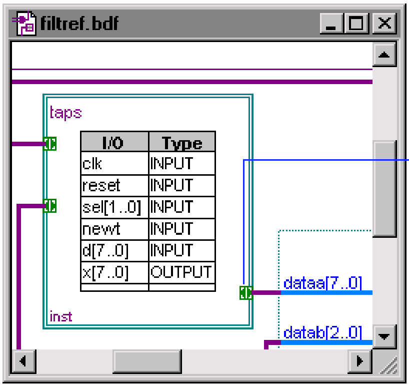
6. Follow steps 1 through 5 to name the node that feeds the Clock input of the `DFF` primitive. Name the node `clk`, so that you can create a logical connection, or a “connection by name,” from the `INPUT` pin `clk` to the Clock input of the `DFF` primitive.

7. Choose **Save** (File menu).

You must map four other connections explicitly so that the Quartus software can properly connect the `mult` symbol to other blocks in the file.

To map other connections between I/O signals in blocks to differently-named I/O signals in other blocks, follow these steps:

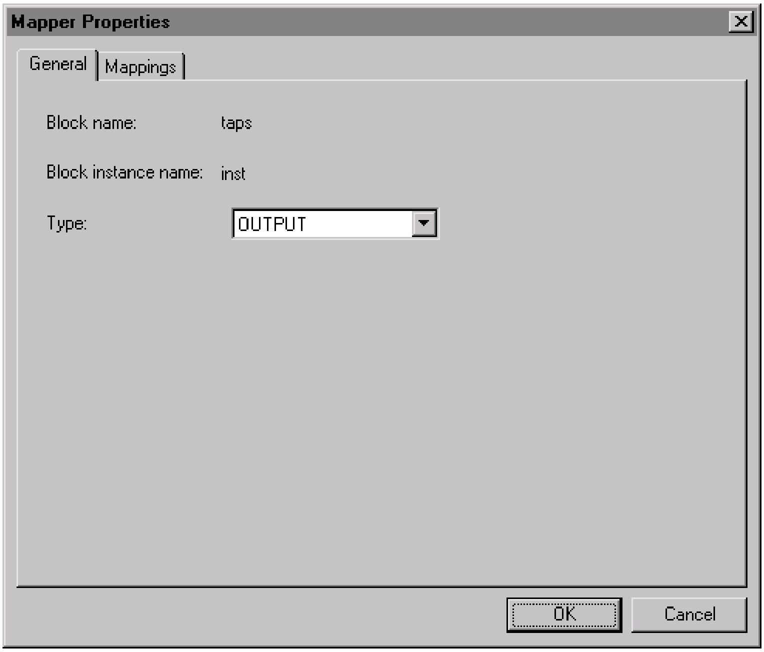
1. On the bus that connects the `taps` block to the `dataa[7..0]` input on the `mult` symbol, double-click the mapper symbol at the end of the bus on the `taps` block, as shown in the following illustration:



I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

When you double-click the mapper symbol, the **Mapper Properties** dialog box appears.

2. In the **Mapper Properties** dialog box, if necessary, click the **General** tab. See the following illustration:



Mapper Properties

Block name: taps

Block instance name: inst

Type: OUTPUT

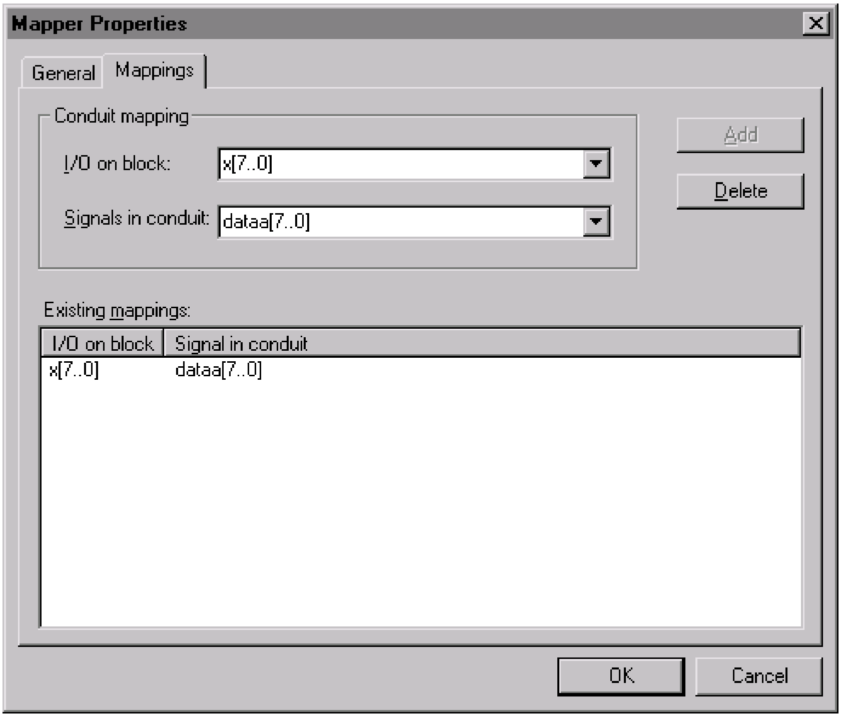
3. In the **Type** list, select **OUTPUT**.

4. Click the **Mappings** tab.

5. In the **I/O on block** list, select **x**[**7..0**].

6. In the **Signals in conduit** box, type `dataa[7..0]`.

7. To map the connection, click **Add**. The mapping appears in the **Existing mappings** list, as shown in the following illustration:



Mapper Properties

General | Mappings

Conduit mapping

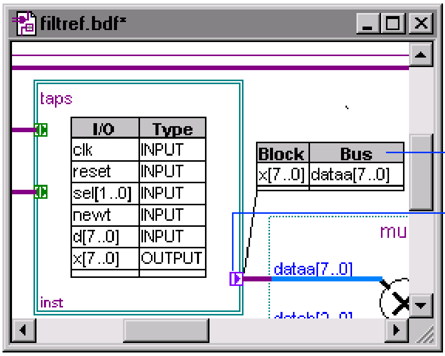
I/O on block: x[7..0]

Signals in conduit: dataa[7..0]

Existing mappings:

I/O on block	Signal in conduit
x[7..0]	dataa[7..0]

8. Click **OK**. The signal `dataa[7..0]` is added to the bus, and a mapper table appears that shows the mapping information. This mapping is necessary to indicate which signals should feed the `dataa[7..0]` input port of the `mult` symbol. See the following illustration:



filtref.bdf*

taps

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

Block	Bus
x[7..0]	dataa[7..0]

Mapper table

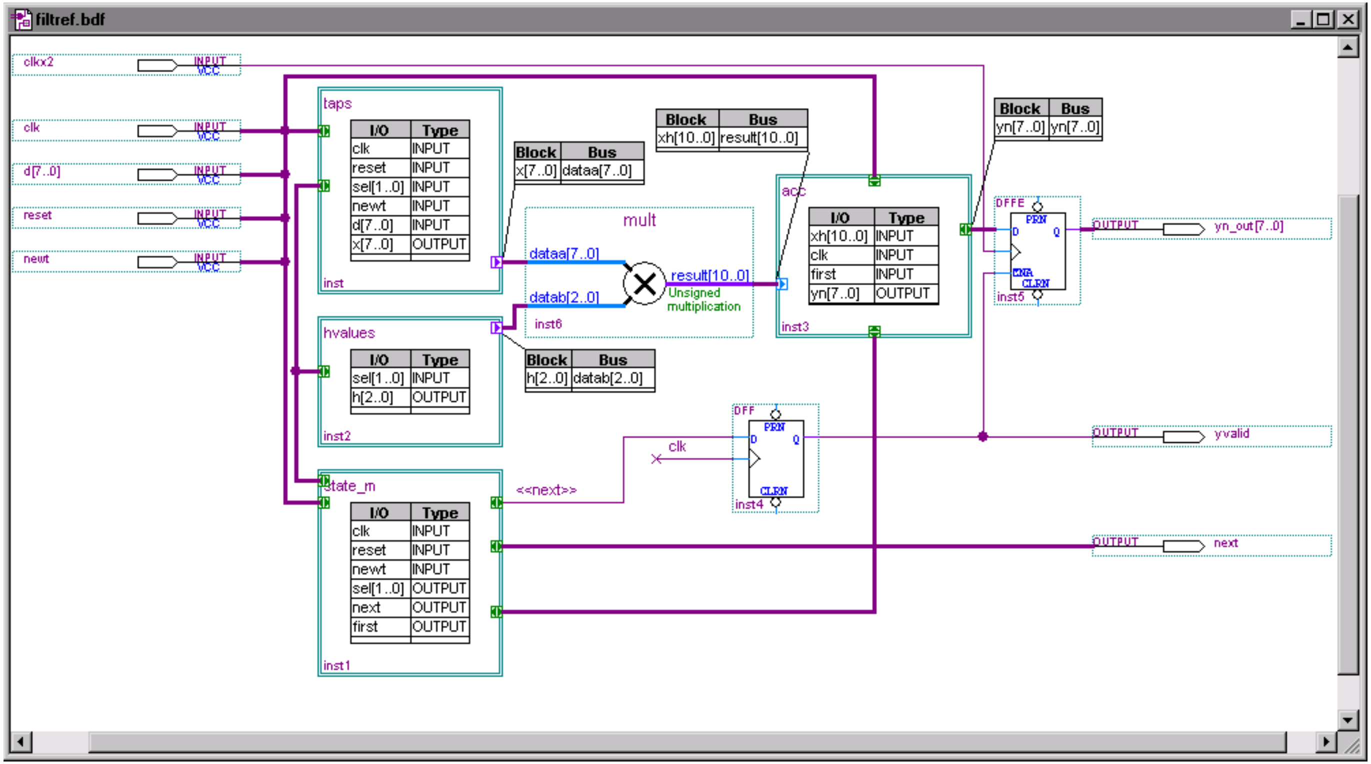
Mapper symbol

9. Repeat steps 1 through 8 to create the following mappings. You can also refer to Figure 2 on page 42, or to the Altera-provided **filtref.bdf** file.

Connection:	**Type:**	**I/O on Block:**	**Signals in Conduit:**
Bus from the `taps` block to the `dataa[7..0]` input of the `mult` symbol (already entered)	`OUTPUT`	`x[7..0]`	`dataa[7..0]`
Bus from the `hvalues` block to the `datab[2..0]` input of the `mult` symbol	`OUTPUT`	`h[2..0]`	`datab[2..0]`
Bus from the `result[10..0]` output of the `mult` symbol to the `acc` block	`INPUT`	`xh[10..0]`	`result[10..0]`
Bus from the `acc` block to the `D` input of the `DFFE` primitive	`BIDIR`	`yn[7..0]`	`yn[7..0]`

10. Choose **Save** (File menu). The Block Design File is complete. See Figure 2 on page 42.

Figure 2. The Completed filtref.bdf Block Design File



taps (inst)

I/O	Type
clk	INPUT
reset	INPUT
sel[1..0]	INPUT
newt	INPUT
d[7..0]	INPUT
x[7..0]	OUTPUT

hvalues (inst2)

I/O	Type
sel[1..0]	INPUT
h[2..0]	OUTPUT

state_m (inst1)

I/O	Type
clk	INPUT
reset	INPUT
newt	INPUT
sel[1..0]	OUTPUT
next	OUTPUT
first	OUTPUT

acc (inst3)

I/O	Type
xh[10..0]	INPUT
clk	INPUT
first	INPUT
yn[7..0]	OUTPUT

Block	Bus
x[7..0]	dataa[7..0]

Block	Bus
h[2..0]	datab[2..0]

Block	Bus
xh[10..0]	result[10..0]

Block	Bus
yn[7..0]	yn[7..0]

Session 4: Create Verilog Design Files

Once you have created a block, you need to create the design file that the block represents, if it does not already exist. You can automatically create a design file that contains the basic framework for the block that represents it. You can then fill in the framework with the design details.

If you are already familiar with Verilog HDL design entry using the Quartus Text Editor, you can copy the Altera-provided **hvalues.v**, **taps.v**, **state_m.v**, and **acc.v** files from the **\qdesigns\tutorial** subdirectory into the **\qdesigns\fir_filter** subdirectory. Altera recommends that you copy tutorial files by opening them in the Quartus software with the **Open** command (File menu), choosing **Save As** (File menu), turning on **Add file to current project**, and then saving the file to the **\qdesigns\fir_filter** subdirectory. If you copy the Altera-provided files, skip to "Session 5: Create a Design File with the MegaWizard Plug-In Manager" on page 47.

This session contains the following steps:

1. Create a Verilog Design File (.v) for the `hvalues` block.
2. Copy Verilog Design Files for other blocks.

1. Create a Verilog Design File for the hvalues Block

To create the framework of a Verilog Design File (**.v**) for the `hvalues` block, follow these steps:

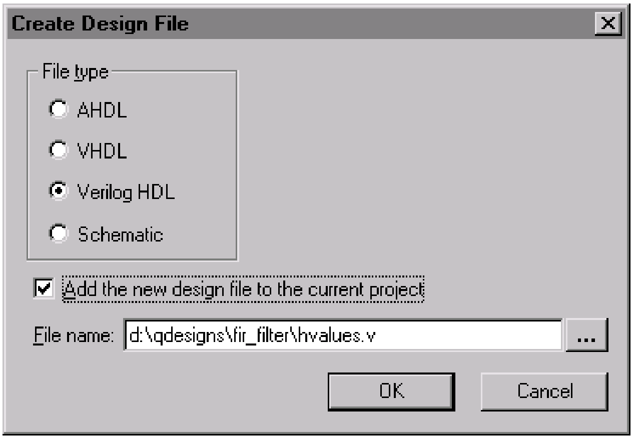
1. Select the `hvalues` block.

2. Choose **Create Design File** (right button pop-up menu).

3. Under **File type**, select **Verilog HDL**.

4. Turn on **Add the new design file to the current project**.

5. Make sure that the **File name** box shows the **hvalues.v** file in the **fir_filter** project directory. See the following illustration:



Create Design File
File type
○ AHDL
○ VHDL
● Verilog HDL
○ Schematic
☑ Add the new design file to the current project
File name: d:\qdesigns\fir_filter\hvalues.v
...
OK
Cancel

6. Click **OK**. The Quartus software confirms that the file has been generated successfully and automatically opens a Text Editor window that contains the new file.

The Quartus software generates the file that is shown in Figure 3 on page 45, which includes a template for a module declaration, with port declarations that correspond to the data you entered in the block:

☞ You may notice a few pairs of Quartus-generated comments that start with "`ALTERA`" and end with "`DO NOT REMOVE THIS LINE!`" The information between these pairs of comments can be updated later by the Quartus software, so you must not enter text between them. However, you can enter other Verilog HDL statements outside of these commented sections.

Figure 3. Excerpt from the Quartus-Generated hvalues.v File

```
//  Module Declaration
module hvalues
(
   // {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
   sel, h
   // {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);

// Port Declaration

   // {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
   input [1:0] sel;
   output [2:0] h;
   // {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

endmodule
```

7. Add the following lines to the **hvalues.v** file to implement the design. Insert these lines just before the `endmodule` statement:

```
reg [2:0]h;

always @(sel)

case (sel)
       2’b 00 : h = 3’b 111;
       2’b 01 : h = 3’b 101;
       2’b 10 : h = 3’b 011;
       2’b 11 : h = 3’b 001;
endcase
```

The **hvalues.v** file should now appear as shown in Figure 4 on page 46.

Figure 4. Excerpt from the Completed hvalues.v File

```
//  Module Declaration
module hvalues
(
   // {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
   sel, h
   // {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);

// Port Declaration

   // {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
   input [1:0] sel;
   output [2:0] h;
   // {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

    reg [2:0]h;

always @(sel)
case (sel)
      2’b 00 : h = 3’b 111;
      2’b 01 : h = 3’b 101;
      2’b 10 : h = 3’b 011;
      2’b 11 : h = 3’b 001;
endcase

endmodule
```

8. Choose **Save** (File menu).

2. Copy Verilog Design Files for Other Blocks

To copy Verilog Design Files for the `taps`, `state_m`, and `acc` blocks:

✓ Copy the **taps.v**, **state_m.v**, and **acc.v** files that are provided in the **\qdesigns\tutorial** directory into the **\qdesigns\fir_filter** directory. Altera recommends that you copy tutorial files by opening them in the Quartus software with the **Open** command (File menu), choosing **Save As** (File menu), turning on **Add file to current project**, and then saving the file to the **\qdesigns\fir_filter** subdirectory.

Session 5: Create a Design File with the MegaWizard Plug-In Manager

The design file **acc.v** requires a 12-bit adder function, which you can implement with a variation of the `lpm_add_sub` function. The **acc.v** file instantiates this function as `accum`. You can create this custom variation with the MegaWizard Plug-In Manager. For additional information about the MegaWizard Plug-In Manager, refer to "3. Enter Primitive & Megafunction Symbols" on page 20.

☞ If you are already familiar with using the Megawizard Plug-In Manager, you can copy the following Altera-provided files from the **\qdesigns\tutorial** directory into the **\qdesigns\fir_filter** directory. Altera recommends that you copy tutorial files by opening them in the Quartus software with the **Open** command (File menu), choosing **Save As** (File menu), turning on **Add file to current project**, and then saving the file to the **\qdesigns\fir_filter** subdirectory. If you copy the Altera-provided files, skip to "Compilation" on page 51.

File Name:	**Description:**
accum.v	Custom Verilog HDL variation of the `lpm_add_sub` function created with the MegaWizard Plug-In Manager.
accum.inc	An ASCII text file, generated automatically by the MegaWizard Plug-In Manager, used by the Compiler to declare Verilog HDL port name and order information.

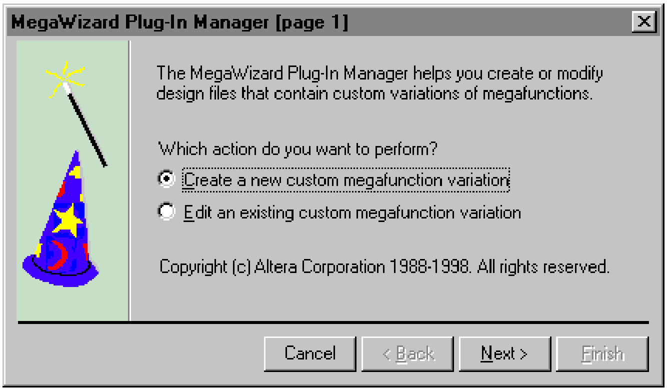
This session includes the following steps:

1. Create a Verilog HDL variation of a megafunction.
2. Add wizard-generated files to the project.

1. Create a Verilog HDL Variation of a Megafunction

To create a custom Verilog HDL version of the `lpm_add_sub` function with the MegaWizard Plug-In Manager, follow these steps:

1. Choose **MegaWizard Plug-In Manager** (Tools menu). The MegaWizard Plug-In Manager appears. See the following illustration:



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

Copyright (c) Altera Corporation 1988-1998. All rights reserved.

2. When you are asked **Which action do you want to perform?**, select **Create a new custom megafunction variation** and click **Next**.

3. In the **Available Megafunctions** list, click the + icon to expand the **arithmetic** folder, and then select **LPM_ADD_SUB**.

4. When you are asked **Which type of output file do you want to create?**, select **Verilog HDL**.

5. When you are asked **What name do you want for the output file?**, specify **d:\qdesigns\fir_filter\accum.v** and click **Next**.

6. When you are asked **How wide should the ‘dataa’ & ‘datab’ input buses be?**, select **8**.

7. When you are asked **Which operating mode do you want for the adder/subtractor?**, select **Addition only** and click **Next**.

8. When you are asked **Is the 'dataa' or 'datab' bus value a constant?**, select **No, both values vary** and click **Next**.

9. When you are asked **Do you want any optional inputs or outputs?**, make sure all of the options are turned off and click **Next.**

10. When you are asked **Do you want to pipeline the function?**, select **No** and click **Next**. A Summary page appears, listing the files that the MegaWizard Plug-In Manager will create.

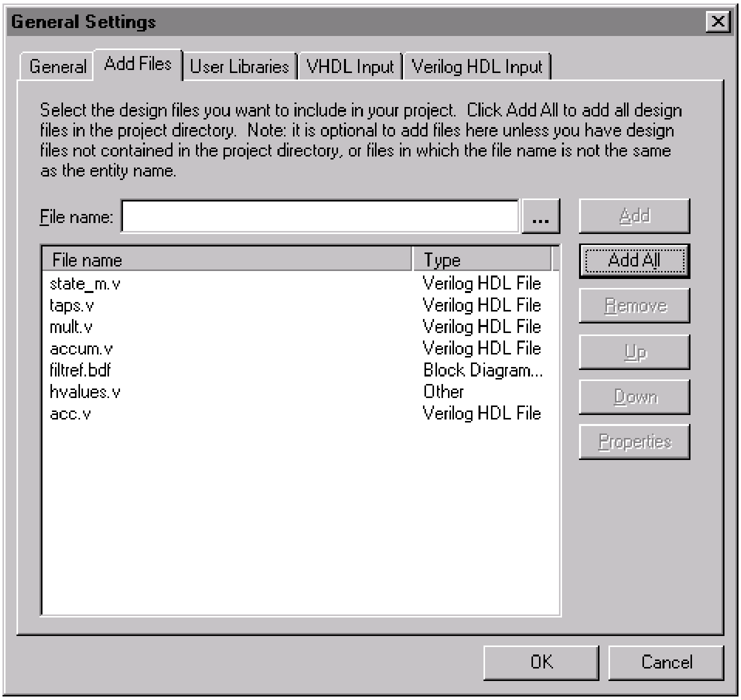
11. Click **Finish**. The wizard creates the **accum.v** file, and the file is ready to be imported into your project.

2. Add Wizard-Generated Files to the Project

To add the **accum.v** and **mult.v** files to the **fir_filter** project, follow these steps:

1. Choose **Add Files to Project** (Project menu). The **Add Files** tab appears automatically.

2. In the **File name** box, specify the **accum.v** file and, if necessary, the **mult.v** file. See the following illustration:



Select the design files you want to include in your project. Click Add All to add all design files in the project directory. Note: it is optional to add files here unless you have design files not contained in the project directory, or files in which the file name is not the same as the entity name.

File name:

File name	Type
state_m.v	Verilog HDL File
taps.v	Verilog HDL File
mult.v	Verilog HDL File
accum.v	Verilog HDL File
filtref.bdf	Block Diagram...
hvalues.v	Other
acc.v	Verilog HDL File

3. Click **OK**. You have now created and added all of the necessary design files for the **fir_filter** project.

Compilation

The Quartus Compiler consists of a series of modules that check the design for errors, synthesize the logic, fit the design into an Altera device, and generate output files for simulation, timing analysis, and device programming.

The Compiler first extracts information that defines the hierarchical connections between a project's design files and checks the designs for basic design entry errors. Then, it creates an organizational map of the design and combines all design files into a fully flattened database that can be processed efficiently.

You can instruct the Compiler to apply a variety of techniques, such as timing-driven compilation, to increase the speed of your design and optimize the device resource usage. The Compiler also creates programming files that the Quartus Programmer or another industry-standard programmer can use to program or configure an Altera device.

Both during and after compilation, you can view the results in the Compilation Report window. The following tutorial sessions guide you through the steps necessary to specify Compiler settings, compile the top-level design entity, view the fit in the Last Compilation floorplan, assign logic to an Embedded System Block (ESB), and recompile the design.

Session 6: Specify Compiler Settings

The Quartus software allows you to compile an entire design, or to compile any constituent part of a design. The "compilation focus," which is the design entity you want to compile, can be selected from any portion of a project's hierarchy.

When you create a new project, the Quartus software creates default Compiler settings that specify the compilation focus, the type of compilation to perform, the device family and device to use, and other options.

You can also create your own customized Compiler settings to override the default settings. You can switch between different Compiler settings when you compile a design. This session includes the following steps:

1. View the Compiler general settings.
2. Specify the device family and device.
3. Specify the Compiler mode.
4. Specify Compiler synthesis and fitting settings.
5. Specify Compiler verification settings.

☞ The procedures below explain how to view and edit Compiler settings using menu commands and dialog boxes. However, you can also easily specify all Compiler settings by following the steps in the **Compiler Settings Wizard** (Processing menu).

1. View the Compiler General Settings

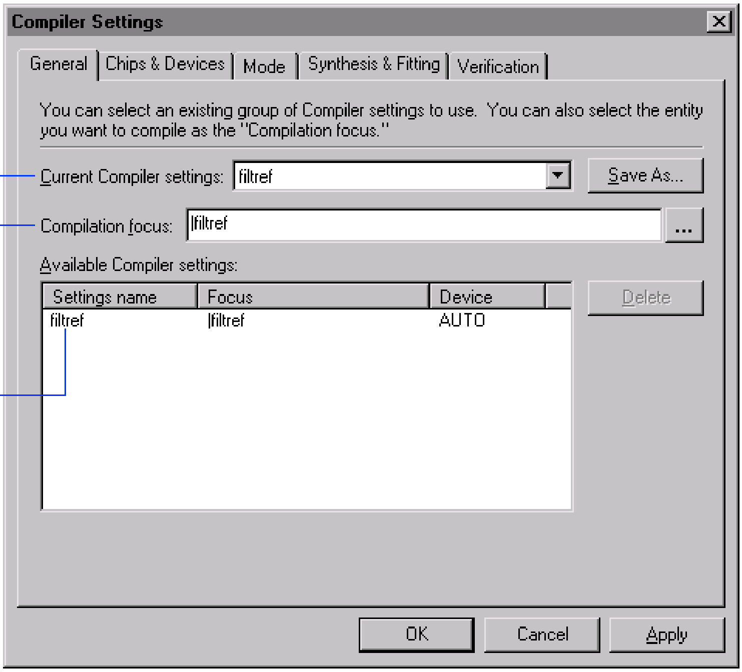
The **General** tab of the **Compiler Settings** dialog box allows you to select an existing group of Compiler settings for use during compilation, define and save a new group of Compiler settings, specify the compilation focus, or delete existing settings.

To view the default Compiler general settings created for the current project, follow these steps:

1. Make sure that you are in Compile mode by selecting **Compile Mode** (Processing menu).

2. Choose **Compiler Settings** (Processing menu). The **General** tab opens automatically.

At this point in the tutorial, the **General** tab displays only the default Compiler general settings created by the Quartus software when you used the **New Project** wizard to create the **fir_filter** project. These default settings are given the name of the top-level design entity in the project, **filtref**. See the following illustration:



You can select an existing group of Compiler settings to use. You can also select the entity you want to compile as the "Compilation focus."

Current Compiler settings: filtref

Compilation focus: |filtref

Available Compiler settings:

Settings name	Focus	Device
filtref	|filtref	AUTO

Specifies the current Compiler settings.

Specifies the hierarchical path name of the design entity you want to compile.

Shows the existing Compiler settings for your project.

2. Specify the Device Family & Device

The **Chips & Devices** tab of the **Compiler Settings** dialog box allows you to select the family and device you want to target for compilation.

To select the device family and device, follow these steps:

1. In the **Compiler Settings** dialog box, click the **Chips & Devices** tab.

2. In the **Family** list, select **APEX20K**.

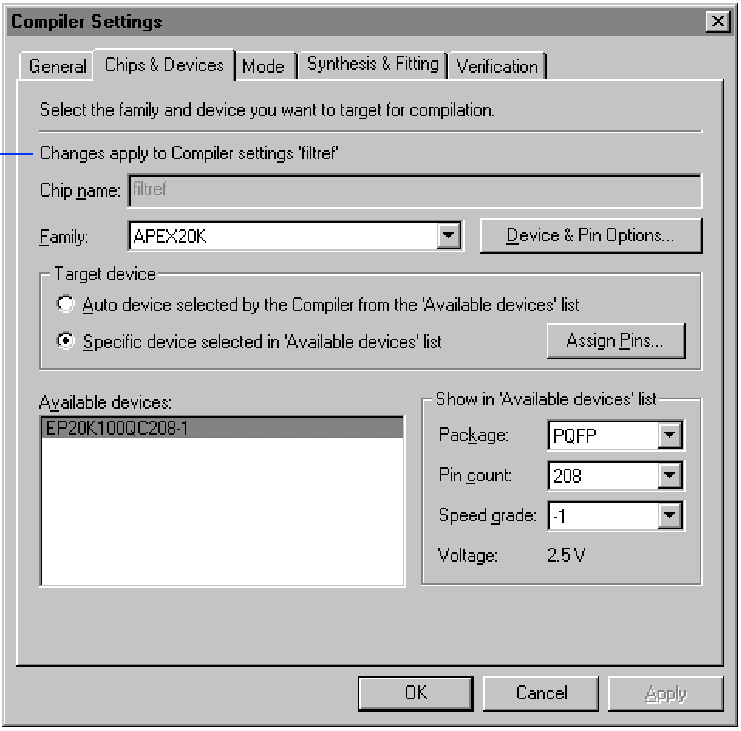
3. Under **Target device**, select **Specific device selected in “Available devices” list**.

4. Under **Show in “Available devices” list**, select the following options:

   a. In the **Package** list, select **PQFP**.

   b. In the **Pin count** list, select **208**.

c. In the **Speed grade** list, select **-1**. See the following illustration:



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

Changes apply to Compiler settings 'filtref'

Identifies the Compiler settings you are editing.

Chip name: filtref

Family: APEX20K

Target device

○ Auto device selected by the Compiler from the 'Available devices' list

● Specific device selected in 'Available devices' list

Available devices: EP20K100QC208-1

Show in 'Available devices' list

Package: PQFP

Pin count: 208

Speed grade: -1

Voltage: 2.5 V

5. In the **Available devices** list, select **EP20K100QC208-1**.

6. Click **Apply**.

3. Specify the Compiler Mode

The **Mode** tab of the **Compiler Settings** dialog box allows you to specify options that affect the type of compilation, the compilation speed, and the amount of disk space used for compilation.

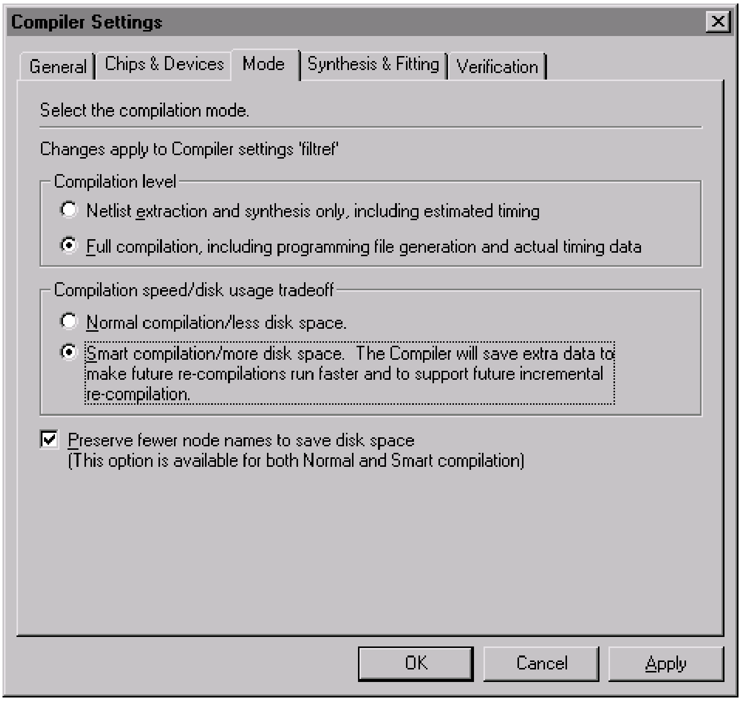
To specify the Compiler mode, follow these steps:

1. In the **Compiler Settings** dialog box, click the **Mode** tab.

2. Under **Compilation level**, select **Full compilation**.

3. To make future re-compilations run faster, under **Compilation speed/disk usage tradeoff**, select **Smart compilation/more disk space**.

4. Make sure the **Preserve fewer node names to save disk space** option is turned on. See the following illustration:



Select the compilation mode.

Changes apply to Compiler settings 'filtref'

Compilation level

○ Netlist extraction and synthesis only, including estimated timing

● Full compilation, including programming file generation and actual timing data

Compilation speed/disk usage tradeoff

○ Normal compilation/less disk space.

● Smart compilation/more disk space. The Compiler will save extra data to make future re-compilations run faster and to support future incremental re-compilation.

☑ Preserve fewer node names to save disk space
(This option is available for both Normal and Smart compilation)

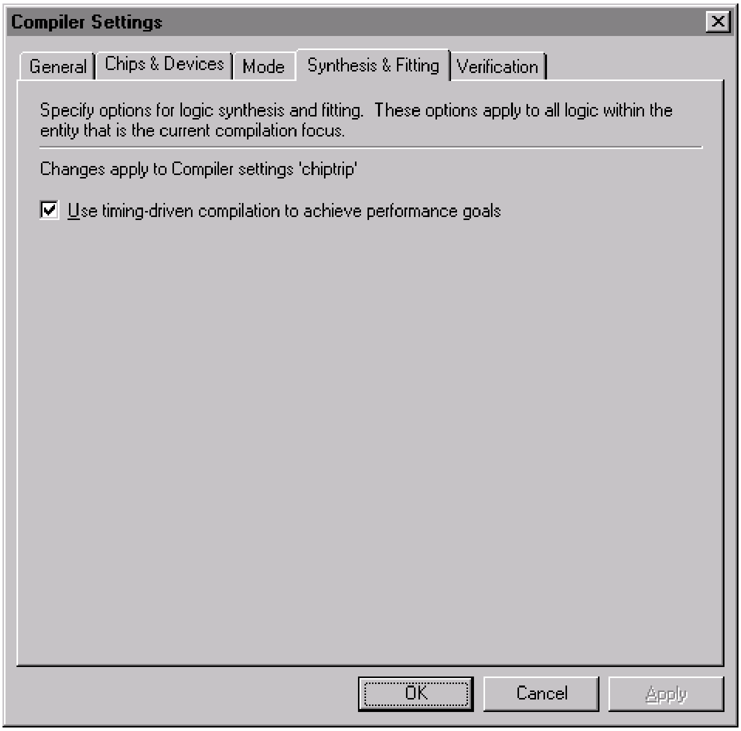
4. Specify Compiler Synthesis & Fitting Settings

The **Synthesis & Fitting** tab of the **Compiler Settings** dialog box allows you to specify options that determine how a design is implemented in a device. You can turn on options in this dialog box to direct the Fitter to optimize the placement of logic in order to meet your timing goals.

To specify Compiler synthesis and fitting settings, follow these steps:

1. In the **Compiler Settings** dialog box, click the **Synthesis & Fitting** tab.

2. Make sure that **Use timing-driven compilation to achieve performance goals** is turned on. See the following illustration:



Specify options for logic synthesis and fitting. These options apply to all logic within the entity that is the current compilation focus.

Changes apply to Compiler settings 'chiptrip'

☑ Use timing-driven compilation to achieve performance goals

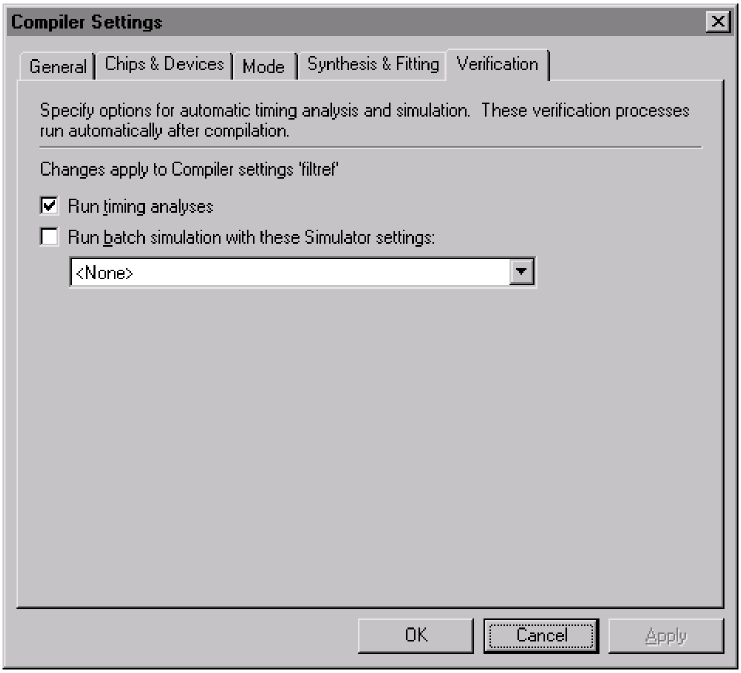
5. Specify Compiler Verification Settings

The **Verification** tab of the **Compiler Settings** dialog box allows you to specify options for running automatic timing analyses and/or a batch simulation at the end of the compilation process.

To specify Compiler verification settings, follow these steps:

1. In the **Compiler Settings** dialog box, click the **Verification** tab.

2. Make sure that **Run timing analyses** is turned on. See the following illustration:



Specify options for automatic timing analysis and simulation. These verification processes run automatically after compilation.

Changes apply to Compiler settings 'filtref'

☑ Run timing analyses

□ Run batch simulation with these Simulator settings:

<None>

3. Click **OK**. All of the settings and options you specified are saved in the **filtref** Compiler settings. When you run the Compiler, these current Compiler settings apply.

Session 7: Compile the Design

During compilation, the currently specified Compiler settings control design processing. The Compiler automatically locates and uses all non-design files associated with the current compilation focus, such as Include Files (**.inc**) containing AHDL Function Prototype Statements; Memory Initialization Files (**.mif**) or Hexadecimal Intel-format Files (**.hex**) containing the initial content of memories; and Project, Entity, and Compiler Settings Files (**.psf**, **.esf**, and **.csf**). During compilation, the Compiler generates information, warning, and error messages that appear automatically in the Messages window.

This session includes the following steps:

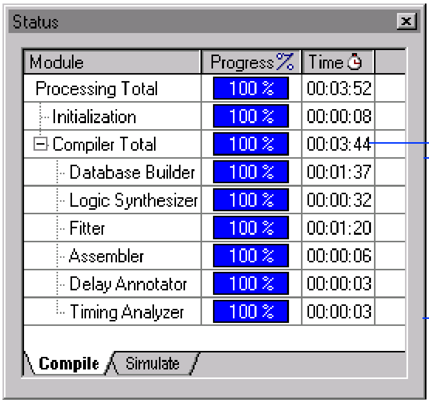
1. Run the Compiler.
2. Locate the source of a message.
3. View the Compilation Report.

1. Run the Compiler

To compile the **filtref** design entity, follow these steps:

1. Choose **Start Compilation** (Processing menu).

   The Compiler immediately begins to compile the **filtref** design entity, and all of its subordinate design entities, using the **filtref** Compiler settings. As the design compiles, the Status window automatically displays, as a percentage, the total compilation progress and the time spent in each stage of the compilation. See the following illustration:



Module	Progress %	Time
Processing Total	100 %	00:03:52
Initialization	100 %	00:00:08
Compiler Total	100 %	00:03:44
Database Builder	100 %	00:01:37
Logic Synthesizer	100 %	00:00:32
Fitter	100 %	00:01:20
Assembler	100 %	00:00:06
Delay Annotator	100 %	00:00:03
Timing Analyzer	100 %	00:00:03

Total compilation time

Time spent in individual modules

In addition, the results of the compilation are updated in the Compilation Report window.

The Compiler runs in the background, freeing your computer—and other portions of the Quartus software—for other work. However, the compilation of this design entity is short and you should not have to wait long for it to finish.

2. When you receive a message indicating that compilation was successful, click **OK**.

2. Locate the Source of a Message

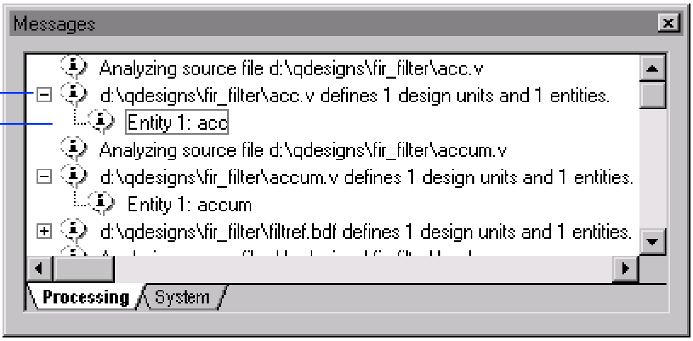
During compilation, all compilation messages appear in the **Processing** tab of the Messages window. Some messages that appear in the Messages window can be located in a design or other source file.

To locate the source of a Compiler-generated message, follow these steps:

1. In the Messages window, click the + icon to expand the `d:\qdesigns\fir_filter\acc.v defines` information message. See the following illustration:

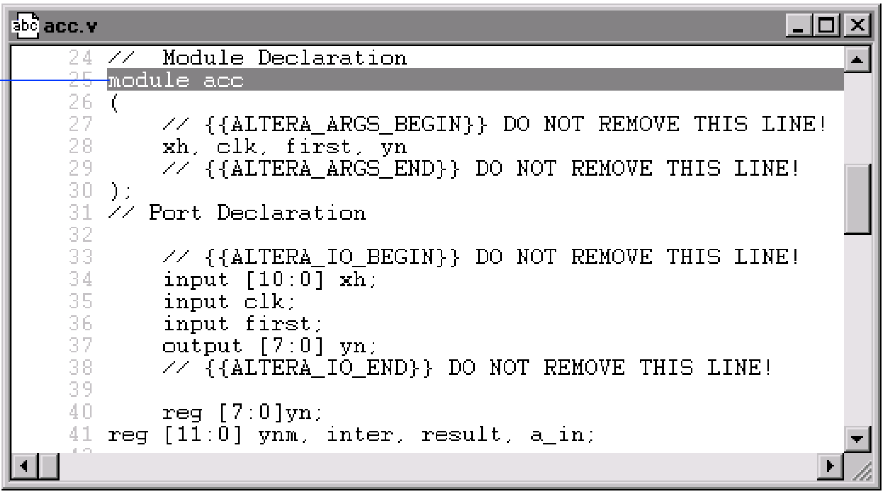
Information message

Expanded information message



```
Analyzing source file d:\qdesigns\fir_filter\acc.v
d:\qdesigns\fir_filter\acc.v defines 1 design units and 1 entities.
    Entity 1: acc
Analyzing source file d:\qdesigns\fir_filter\accum.v
d:\qdesigns\fir_filter\accum.v defines 1 design units and 1 entities.
    Entity 1: accum
d:\qdesigns\fir_filter\filtref.bdf defines 1 design units and 1 entities.
```

2. Double-click the `Entity 1:acc` expanded information message. The **acc.v** file, which contains the message source, appears in a Text Editor window, and the section that is the source of the message—the Module Declaration—is highlighted in the file. See the following illustration:



```
//  Module Declaration
module acc
(
    // {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
    xh, clk, first, yn
    // {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);
// Port Declaration

    // {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
    input [10:0] xh;
    input clk;
    input first;
    output [7:0] yn;
    // {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

    reg [7:0]yn;
reg [11:0] ynm, inter, result, a_in;
```

Source of message is highlighted in the original file.

3. Once you have finished viewing the source of the message, close the Text Editor window.

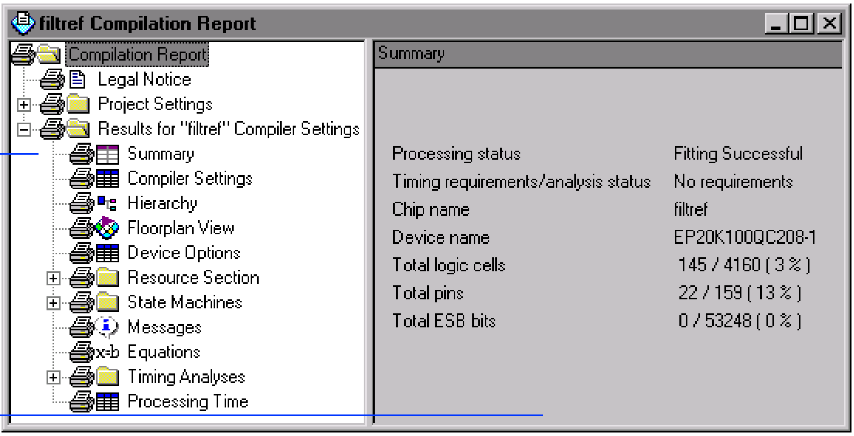
3. View the Compilation Report

During compilation, the Compilation Report window appears automatically. The Compilation Report provides detailed information about the current compilation. When the report first opens, the Summary section appears in the right pane of the window. This section provides summary information about the compilation, including its final status; timing requirements, if any; the name of the design entity compiled; the total number of logic cells and pins used in the device; and the total amount of memory used.

The Compilation Report's other sections provide information about the design hierarchy revealed by the compilation; the floorplan showing how the Compiler placed logic in the device; pin and logic cell usage; global and control signal usage; device interconnect usage; operating frequency ($\mathbf{f}_{\mathrm{MAX}}$); setup and hold time, and pin-to-pin timing; equations; processing time; etc.

To view any section of the Compilation Report, follow these steps:

1. In the left pane of the Compilation Report window, click the + icon to expand the appropriate Report window section. See the following illustration:



Processing status	Fitting Successful
Timing requirements/analysis status	No requirements
Chip name	filtref
Device name	EP20K100QC208-1
Total logic cells	145 / 4160 (3 %)
Total pins	22 / 159 (13 %)
Total ESB bits	0 / 53248 (0 %)

Left pane selects the Compilation Report section.

Right pane shows the Compilation Report section.

2. In the left pane of the Report window, select the Report section you want to open. The report appears in the right pane of the window.

Go to "Session 10: View Timing Analysis Results" on page 73 for more information about timing analysis results. Go to "Overview: Viewing the Results of a Compilation or Simulation in the Report Window" in Quartus Help for more information about the Report window.

Session 8: View the Fit in the Last Compilation Floorplan

The Quartus software offers two views of a design's floorplan:

- The Current Assignments floorplan allows you to edit the location assignments of resources on the device.
- The non-editable Last Compilation floorplan shows how the Compiler implemented the design in a device.

Each of these floorplans allows you to view information organized by interior logic cells, interior logic array blocks (LABs), interior MegaLAB™ structures, and the device package top and bottom.

☞ The actual placement and routing of logic in the device may differ from that shown in the floorplan illustrations in this manual due to improvements made in the Quartus software after this manual was printed.

This session includes the following steps:

1. Open the Last Compilation floorplan.
2. Display routing information.
3. Display equation information.
4. Display the MegaLAB view.

1. Open the Last Compilation Floorplan

After a successful compilation, you can view the results in the Last Compilation floorplan. The Last Compilation floorplan shows how the Compiler implemented the logic of a design into an Altera device.

To open the Last Compilation floorplan, follow these steps:

1. Choose **Open Last Compilation Floorplan** (Processing menu).

2. To increase the size of the Floorplan Editor window, turn on **Full Screen** (View menu). By default, the Interior Logic Cell view of the Floorplan Editor appears, showing the individual logic cells in each LAB. Unused logic cells are shown in white. Used logic cells are color-coded to reflect the interconnect used in fitting.

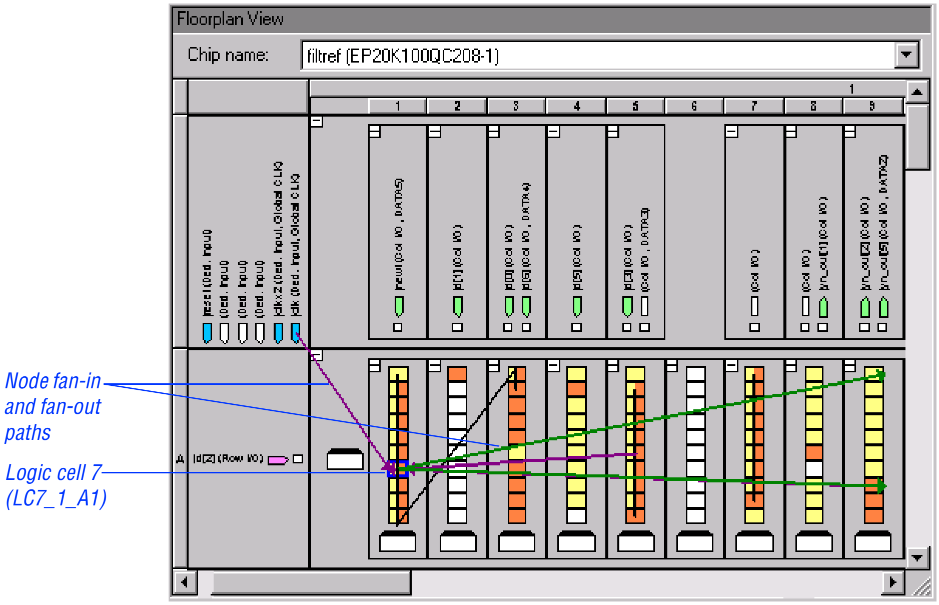
3. To display the color legend, turn on **Color Legend Window** (View menu).

2. Display Routing Information

To show node fan-in and fan-out, follow these steps:

1. Select **Routing > Show Node Fan-In & Fan-Out** (View menu).

2. In the first LAB (LAB1), select the seventh logic cell down (LC7). The fan-in to the logic cell is displayed in dark magenta; the fan-out is shown in dark cyan. The arrows on the fan-in and fan-out lines indicate the direction of flow for the signal. See the following illustration:



Chip name: filtref (EP20K100QC208-1)

Node fan-in and fan-out paths

Logic cell 7 (LC7_1_A1)

3. To display the routing delays, turn on **Routing > Show Routing Delays** (View menu). The routing delays associated with the selected logic cell appear in the floorplan alongside the fan-in and fan-out paths.

4. When you are finished viewing the fan-in and fan-out paths, select **Routing > Hide Routing** (View menu).

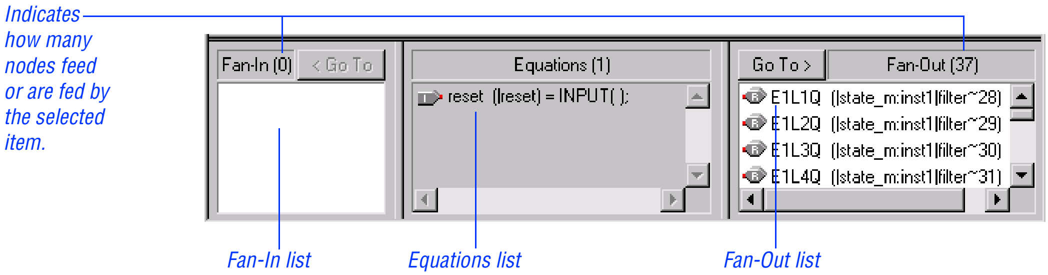
3. Display Equation Information

The Equations window allows you to view the equations, fan-in, and fan-out for a selected pin or cell. The Equations window appears automatically at the bottom of the Floorplan Editor window.

To view the equations for the `reset` input pin, follow these steps:

1. If necessary, to display the Equations window, turn on **Equations** (View menu).

2. Select the `reset` input pin in the top left-hand corner of the Last Compilation floorplan. The Equations window displays the equation for the selected pin. The **Fan-Out** list displays the signals fed by the logic cell. See the following illustration:



```
reset (|reset) = INPUT( );
```

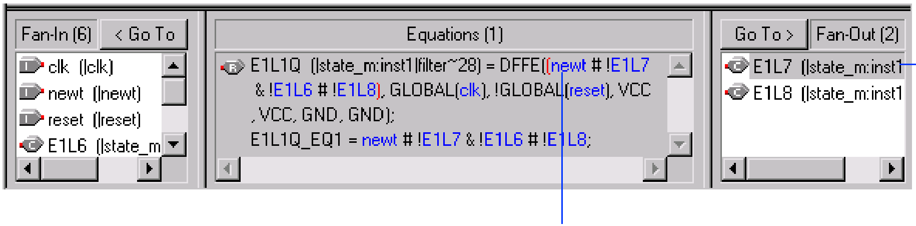
```
E1L1Q (|state_m:inst1|filter~28)
E1L2Q (|state_m:inst1|filter~29)
E1L3Q (|state_m:inst1|filter~30)
E1L4Q (|state_m:inst1|filter~31)
```

3. To expand the equation associated with the `E1L1Q(|state_m:inst1|filter~28)` node, follow these steps:

   a. At the top of the **Fan-Out** list, select the `E1L1Q(|state_m:inst1|filter~28)` registered node name.

   b. Click **Go To** >. The equation for the register appears in the **Equations** list **.**

   c. In the **Equations** list, click the first equation term highlighted in blue. The equation term expands to reveal the substituted equation, as shown in the following illustration:



clk (|clk)
newt (|newt)
reset (|reset)
E1L6 (|state_m

```
E1L1Q (|state_m:inst1|filter~28) = DFFE((newt # !E1L7
   & !E1L6 # !E1L8), GLOBAL(clk), !GLOBAL(reset), VCC
   ,VCC, GND, GND);
E1L1Q_EQ1 = newt # !E1L7 & !E1L6 # !E1L8;
```

E1L7 (|state_m:inst1
E1L8 (|state_m:inst1

Selected node

Expanded equation

4. Display the MegaLAB View

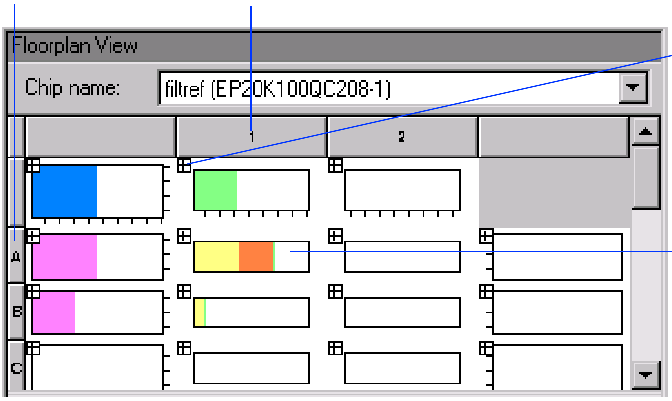
MegaLABs are large-scale structures within APEX 20K™ devices. Each MegaLAB contains 16 LABs, one ESB, and a MegaLAB Interconnect structure.

To display the MegaLABs of the device in the Floorplan Editor window, follow these steps:

1. Select **Interior MegaLABs** (View menu). The Floorplan Editor displays the MegaLABs in the device. Unused resources are shown in white, by default. Used resources are color-coded to reflect the interconnect used in fitting. See the following illustration:

Row A

Column 1



Chip name: filtref (EP20K100QC208-1)

Clicking the + icon expands the MegaLAB to show details.

MegaLAB_A1

2. In column 1 row A, click the + icon to expand **MegaLAB_A1.** The MegaLAB expands to show its individual LABs.

3. Turn off **Full Screen** (View menu).

4. Select **Interior Cells** (View menu).

5. To close the Last Compilation floorplan, choose **Close** (File menu).

Session 9: Assign Logic to an ESB

You can use a technology mapping option to force the Quartus software to implement the logic of a design entity into ESBs. ESBs are structures within APEX 20K devices that can be used to implement memory (RAM or ROM), or to implement registered logic in the ESB's product term macrocells.

This session includes the following steps:

1. Back-annotate assignments.
2. Verify the back-annotated assignments.
3. Create a location assignment.
4. Recompile the design.
5. View the fit in the Last Compilation floorplan.

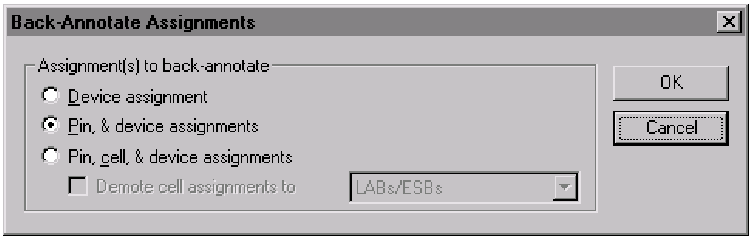
1. Back-Annotate Assignments

After you have compiled the design, you can use back-annotation to preserve the resource assignments that the Compiler made during the most recent compilation. Back-annotation allows you to achieve the same fit with subsequent compilations.

To back-annotate the pin and device assignments, follow these steps:

1. Choose **Back-Annotate Assignments** (Processing menu).

2. In the **Back-Annotate Assignments** dialog box, under **Assignment(s) to back-annotate**, select **Pin & device assignments**. See the following illustration:



Back-Annotate Assignments

Assignment(s) to back-annotate

○ Device assignment

● Pin, & device assignments

○ Pin, cell, & device assignments

□ Demote cell assignments to LABs/ESBs

3. Click **OK**.

2. Verify the Back-Annotated Assignments

You can verify that back-annotation preserved the last compilation assignments by viewing the Current Assignments floorplan.

To verify the assignments, follow these steps:

1. Choose **Open Current Assignments Floorplan** (Processing menu).

2. Turn on **Full Screen** (View menu). The **Chip name** list at the top of the Current Assignments floorplan displays the chip name and EP20K100QC208-1 device you specified, and the pin assignments from the last compilation are now shown in the Current Assignments floorplan. Because the back-annotation did not include logic cells, all logic cells are empty.

3. Turn off **Full Screen** (View menu).

4. To close the Floorplan Editor, choose **Close** (File menu).

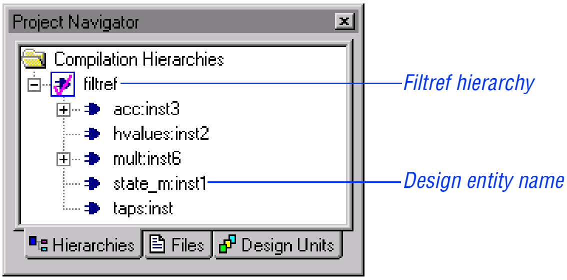
3. Create a Location Assignment

To assign the `state_m:inst1` design entity to an ESB product term, follow these steps:

1. In the Project Navigator, click the **Hierarchies** tab.

☞ The Project Navigator appears by default when you start the Quartus software. However, if necessary, you can display the Project Navigator by turning on **Auxiliary Windows > Project Navigator** (View menu).

2. In the **Hierarchies** tab, click the + icon to expand the **filtref** hierarchy. See the following illustration:



3. In the **filtref** hierarchy, select the `state_m:inst1` design entity name.

4. Choose **Assignment Organizer** (right button pop-up menu). The **Assignment Organizer** dialog box opens, with the **Edit specific entity & node settings for** option selected, and with the hierarchical path name of the `state_m:inst1` entity shown in the **Name** box.

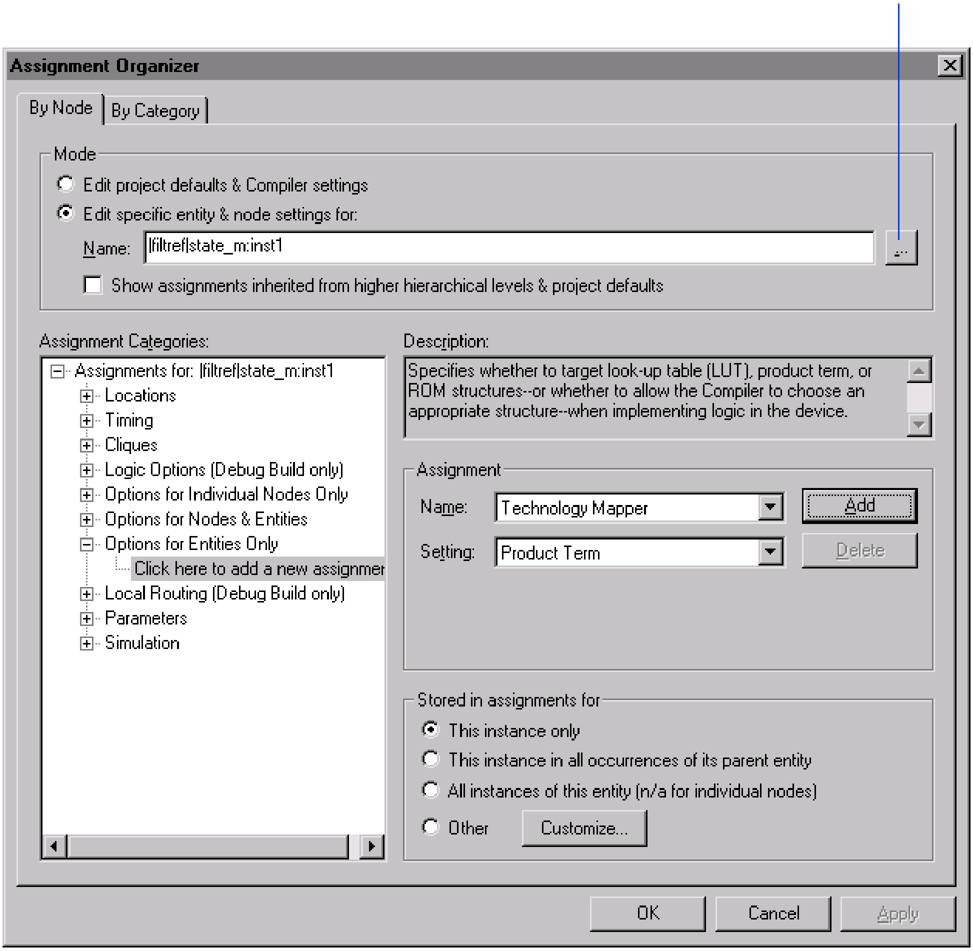
5. In the **Assignment Categories** list, click the + icon to expand **Options for Entities Only**.

6. Click the **Click here to add a new assignment** text.

7. Under **Assignment**, in the **Name** list, select **Technology Mapper**.

8. In the **Setting** list, select **Product Term**. See the following illustration:

Browse button opens the Node Finder.



Assignment Organizer

By Node | By Category

Mode

○ Edit project defaults & Compiler settings

● Edit specific entity & node settings for:

Name: |filtref|state_m:inst1

□ Show assignments inherited from higher hierarchical levels & project defaults

Assignment Categories:

- Assignments for: |filtref|state_m:inst1
  - Locations
  - Timing
  - Cliques
  - Logic Options (Debug Build only)
  - Options for Individual Nodes Only
  - Options for Nodes & Entities
  - Options for Entities Only
    - Click here to add a new assignmer
  - Local Routing (Debug Build only)
  - Parameters
  - Simulation

Description:

Specifies whether to target look-up table (LUT), product term, or ROM structures--or whether to allow the Compiler to choose an appropriate structure--when implementing logic in the device.

Assignment

Name: Technology Mapper

Setting: Product Term

Stored in assignments for

● This instance only

○ This instance in all occurrences of its parent entity

○ All instances of this entity (n/a for individual nodes)

○ Other

9. Click **Add**. The assignment appears in the **Assignment Categories** list.

10. Click **OK**.

4. Recompile the Design

To recompile the design and implement your new assignment:

✓ Choose **Start Compilation** (Processing menu).

5. View the Fit in the Last Compilation Floorplan

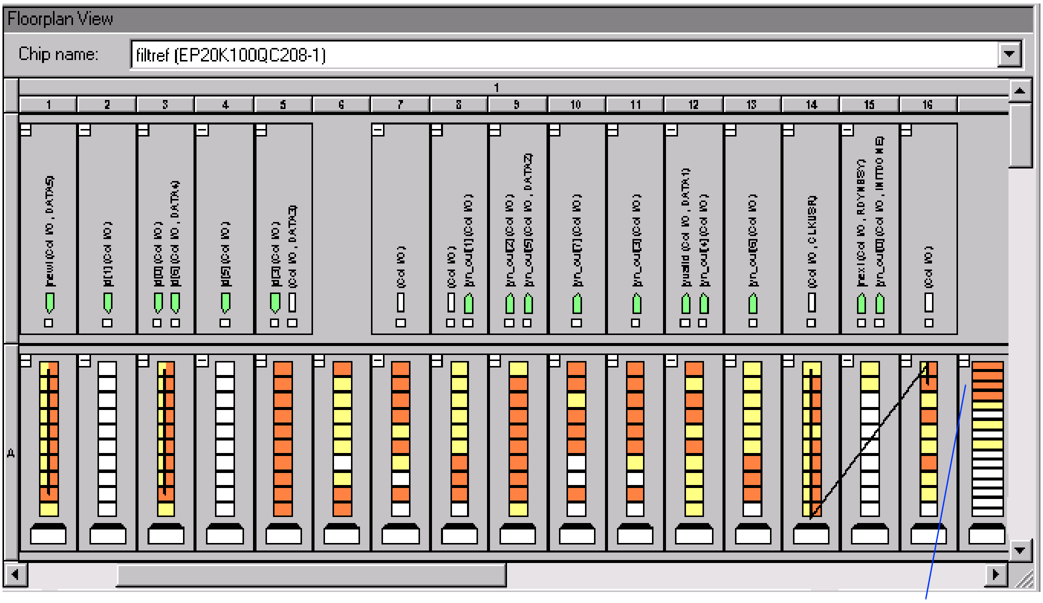
To view the fit in the Last Compilation floorplan, follow these steps:

1. Choose **Open Last Compilation Floorplan** (Processing menu).

2. To increase the size of the Floorplan Editor window, turn on **Full Screen** (View menu). Note that the pin-out has been preserved from the previous fit and that logic has been implemented in ESBs.

3. If necessary, click the **Selection Tool** button on the Floorplan Editor toolbar.

4. With the Selection Tool, point to some of the used ESBs, located at the center of row 1. The names of the signals appear in "bubble text." The signal names all start with "`state_m`," indicating that they are part of the `state_m` entity. See the following illustration:



Floorplan View

Chip name: filtref (EP20K100QC208-1)

ESBs

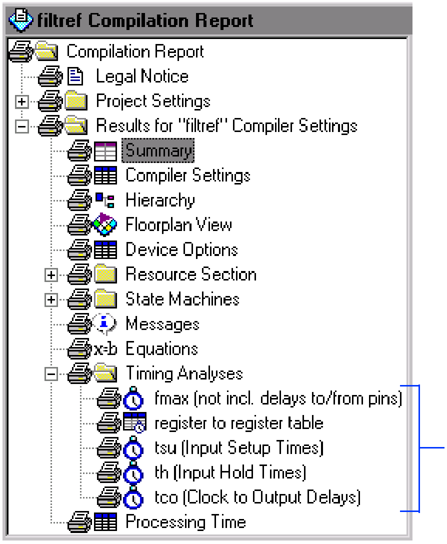
5. Turn off **Full Screen** (View menu).

Timing Analysis

With the Quartus Timing Analyzer, you can analyze the timing performance of all logic that has been processed by the Compiler. You can trace signal paths and locate them in the Floorplan Editor, determining critical speed paths that limit the design's performance.

The Quartus Timing Analyzer runs automatically at the end of the compilation process, or you can run the Timing Analyzer separately after first-time project compilation. By default, the Timing Analyzer reports the maximum frequency ($\mathbf{f}_{\mathrm{MAX}}$) of every register, the worst-case register to register delays, the input setup ($\mathbf{t}_{\mathrm{SU}}$) and input hold ($\mathbf{t}_{\mathrm{H}}$) times of every input register, and the clock-to-output ($\mathbf{t}_{\mathrm{CO}}$) delays of every output register in the design hierarchy. In addition, the Timing Analyzer analyzes all pin-to-pin paths and reports the pin-to-pin ($\mathbf{t}_{\mathrm{PD}}$) delays between pins.

When compilation is complete, timing information is reported in the **Timing Analyses** folder of the Compilation Report, as shown in the following illustration:



Timing Analyses folder

The following tutorial sessions guide you through the steps necessary to view the timing analysis results in the Compilation Report window, specify timing requirements, perform multi-clock timing analysis, and assign a multicycle path.

Session 10: View Timing Analysis Results

After you compile a design, you can view the timing analysis results in the Compilation Report. This session includes the following steps:

1. View the f_{MAX} timing analysis report.
2. List the f_{MAX} timing paths.
3. Locate an f_{MAX} timing path in the Floorplan Editor.
4. View the t_{SU} timing analysis report.

Go to "Viewing Timing Analysis Results in the Report Window" in Quartus Help for specific steps on viewing other timing analysis results.

1. View the f_{MAX} Timing Analysis Report

The **fmax** section of the Compilation Report shows the frequency requirements and worst-case speed performance of your design. The **fmax** section title indicates whether the f_{MAX} calculations include delays to and from device pins, according to the options you select in the **Clock Settings** tab of the **Timing Settings** dialog box (Project menu).

To open the **fmax** section of the Compilation Report, follow these steps:

1. If necessary, to open the Compilation Report, choose **Open Compilation Report** (Processing menu).

2. In the left pane of the Compilation Report window, click the + icon to expand the **Timing Analyses** folder.

3. Under the **Timing Analyses** folder, select the **fmax** section. The f_{MAX} information appears in a table. If you specify one or more timing requirements related to the current section, the "actual" values appear in red if your timing requirements have not been met.

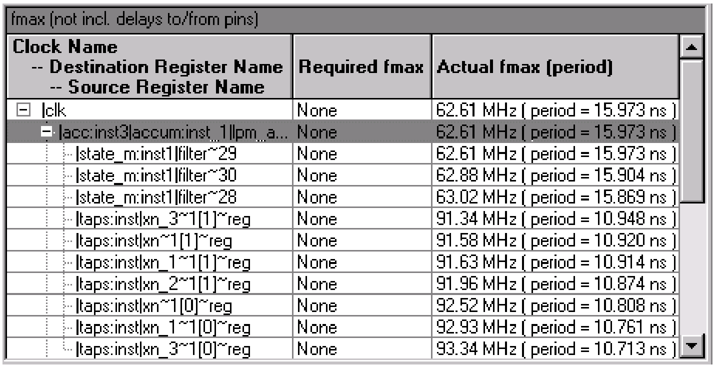
4. To expand the list and display the destination registers associated with the clock, click the + icon next to the `clk` clock name. By default, the list expands to show the 10 slowest destination registers.

Go to "Specifying Timing Analysis Reporting Restrictions" in Quartus Help for information about customizing timing analysis reporting.

2. List the f_{MAX} Timing Paths

To list the f_{MAX} timing paths, follow these steps:

1. To expand the list and display the source registers feeding a destination register, click the + icon to expand the first destination register name in the **fmax** section. By default, the list expands to show the 10 slowest source registers. See the following illustration:



fmax (not incl. delays to/from pins)

<table>
<tr><th>Clock Name
-- Destination Register Name
-- Source Register Name</th><th>Required fmax</th><th>Actual fmax (period)</th></tr>
<tr><td>⊟ |clk</td><td>None</td><td>62.61 MHz (period = 15.973 ns)</td></tr>
<tr><td>⊟ |acc:inst3|accum:inst_1|lpm_a...</td><td>None</td><td>62.61 MHz (period = 15.973 ns)</td></tr>
<tr><td>|state_m:inst1|filter~29</td><td>None</td><td>62.61 MHz (period = 15.973 ns)</td></tr>
<tr><td>|state_m:inst1|filter~30</td><td>None</td><td>62.88 MHz (period = 15.904 ns)</td></tr>
<tr><td>|state_m:inst1|filter~28</td><td>None</td><td>63.02 MHz (period = 15.869 ns)</td></tr>
<tr><td>|taps:inst|xn_3~1[1]~reg</td><td>None</td><td>91.34 MHz (period = 10.948 ns)</td></tr>
<tr><td>|taps:inst|xn~1[1]~reg</td><td>None</td><td>91.58 MHz (period = 10.920 ns)</td></tr>
<tr><td>|taps:inst|xn_1~1[1]~reg</td><td>None</td><td>91.63 MHz (period = 10.914 ns)</td></tr>
<tr><td>|taps:inst|xn_2~1[1]~reg</td><td>None</td><td>91.96 MHz (period = 10.874 ns)</td></tr>
<tr><td>|taps:inst|xn~1[0]~reg</td><td>None</td><td>92.52 MHz (period = 10.808 ns)</td></tr>
<tr><td>|taps:inst|xn_1~1[0]~reg</td><td>None</td><td>92.93 MHz (period = 10.761 ns)</td></tr>
<tr><td>|taps:inst|xn_3~1[0]~reg</td><td>None</td><td>93.34 MHz (period = 10.713 ns)</td></tr>
</table>

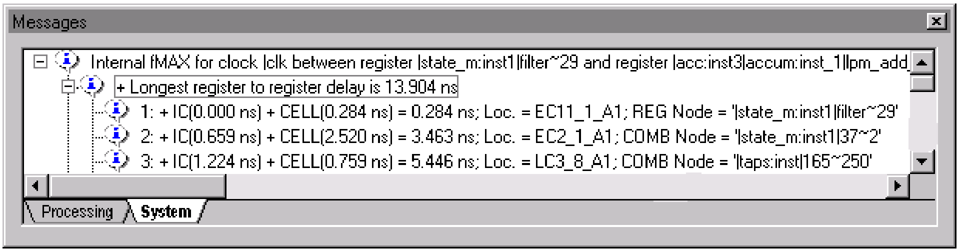
The actual source and/or destination register names may differ from those shown in the previous illustration due to improvements made in the Quartus software after this manual was printed.

2. Select the first source register name in the list.

3. Choose **List Paths** (right button pop-up menu). The delay paths for the source register, including intermediate delay paths, appear as messages in the **System** tab of the Messages window.

4. In the Messages window, click the + icon to expand the `Internal fmax` message. The `Longest register to register delay`, `Smallest Clock skew`, `Micro clock to output delay`, and `Micro setup delay` messages are displayed.

5. Click the + icon to expand the `Longest register to register delay` message. The intermediate time increments used to calculate the timing path delay are displayed. The following illustration shows a sample delay message. The actual values in the message may vary.



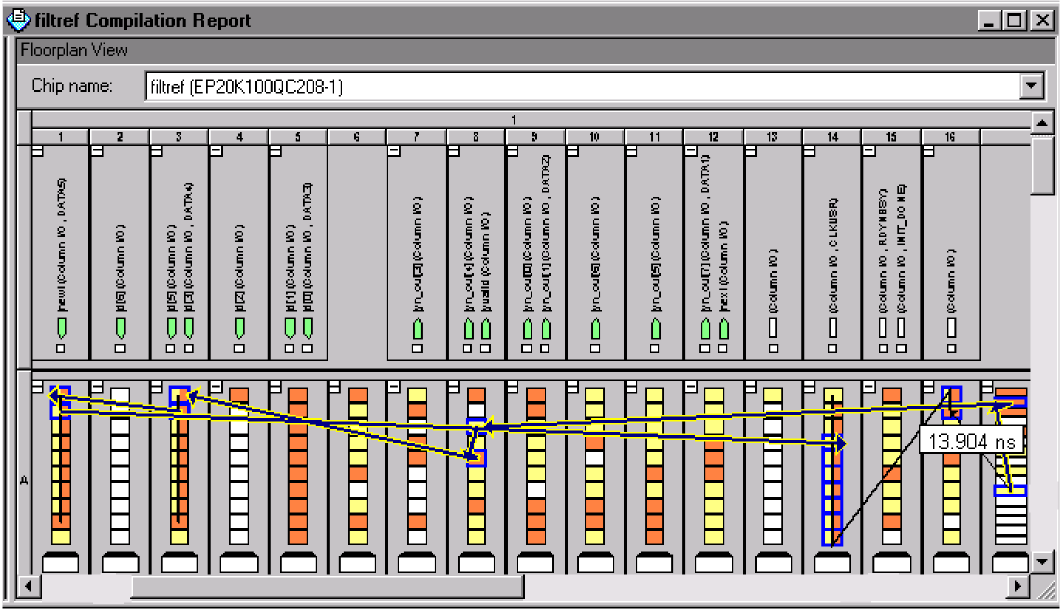
```
Internal fMAX for clock |clk between register |state_m:inst1|filter~29 and register |acc:inst3|accum:inst_1|lpm_add
  + Longest register to register delay is 13.904 ns
    1: + IC(0.000 ns) + CELL(0.284 ns) = 0.284 ns; Loc. = EC11_1_A1; REG Node = '|state_m:inst1|filter~29'
    2: + IC(0.659 ns) + CELL(2.520 ns) = 3.463 ns; Loc. = EC2_1_A1; COMB Node = '|state_m:inst1|37~2'
    3: + IC(1.224 ns) + CELL(0.759 ns) = 5.446 ns; Loc. = LC3_8_A1; COMB Node = '|taps:inst|165~250'
```

3. Locate an f_{MAX} Timing Path in the Floorplan Editor

To locate the source of a timing path message in the Last Compilation floorplan, follow these steps:

1. In the Messages window, select the `Longest register to register` message.

2. Choose **Locate** (right button pop-up menu). The data path is highlighted in the floorplan, and the total timing path delay appears alongside the path. See the following illustration:



filtref Compilation Report

Floorplan View

Chip name: filtref (EP20K100QC208-1)

The actual data path and delay may differ from the previous illustration due to improvements made in the Quartus software after this manual was printed.

3. To close the Last Compilation floorplan, choose **Close** (File menu).

4. View the t_{SU} Timing Analysis Report

The **tsu** section of the Compilation Report displays any setup time requirements and the actual t_{SU} for the input pins that feed the data or clock enable inputs to the destination flipflop.

To open the **tsu** section of the Compilation Report, follow these steps:

1. In the left pane of the Compilation Report window, click the + icon to expand the **Timing Analyses** folder.

2. Under the **Timing Analyses** folder, select the **tsu** section. The input setup time information is displayed in a table.

3. To expand the list and display the register and clock name(s) associated with the data pin, click the + icon.

4. To return to the **filtref.bdf** block diagram, close the Compilation Report window.

☞ You can also view the Register-to-Register, **th**, **tco**, and **tpd** sections in the Compilation Report by modifying the procedures above. Go to “Viewing Timing Analysis Results in the Report Window” in Quartus Help for more information.

Session 11: Specify Timing Requirements

In the Quartus software, you can specify timing requirements and options that apply to the entire project, to specific design entities, or to individual nodes and pins. For example, you can specify timing requirements for the setup time ($\mathbf{t_{SU}}$), hold time ($\mathbf{t_{H}}$), clock-to-output delay ($\mathbf{t_{CO}}$), pin-to-pin delays ($\mathbf{t_{PD}}$), and maximum frequency ($\mathbf{f_{MAX}}$).

You can specify timing requirements for the entire project with the **Timing Settings** command (Project menu). You can specify timing requirements for individual entities, nodes, and pins with the **Assignment Organizer** command (Tools menu).

This session includes the following steps:

1. Specify the default required $\mathbf{f_{MAX}}$.
2. Cut timing paths.

☞ The procedures below explain how to specify timing requirements and other timing analysis settings using menu commands and dialog boxes. However, you can also easily specify timing requirements and other timing analysis settings by following the steps in the **Timing Wizard** (Project menu).

1. Specify the Default Required f_{MAX}

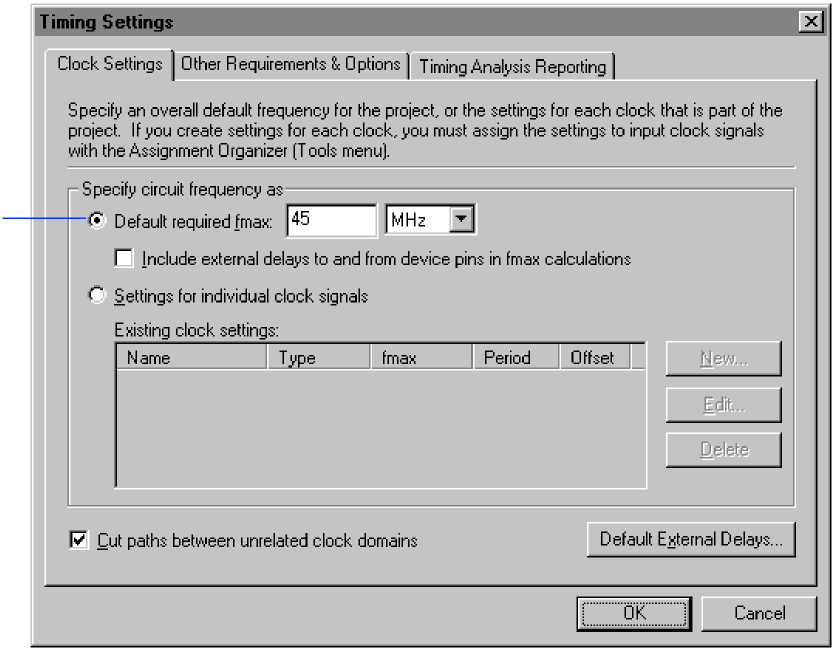
You can specify a target circuit frequency with the $\mathbf{f_{MAX}}$ timing setting. You can specify a default $\mathbf{f_{MAX}}$ for an entire project, or you can specify a required $\mathbf{f_{MAX}}$ for selected portions of the project.

To specify the default required f_{MAX}, follow these steps:

1. Choose **Timing Settings** (Project menu). The **Clock Settings** tab opens automatically.

2. Under **Specify circuit frequency as**, select **Default required fmax**.

3. In the **Default required fmax** box, type 45 and select **MHz** in the list. See the following illustration:



Timing Settings

Specify an overall default frequency for the project, or the settings for each clock that is part of the project. If you create settings for each clock, you must assign the settings to input clock signals with the Assignment Organizer (Tools menu).

Specify circuit frequency as

● Default required fmax: 45 MHz

□ Include external delays to and from device pins in fmax calculations

○ Settings for individual clock signals

Existing clock settings:

Name	Type	fmax	Period	Offset

☑ Cut paths between unrelated clock domains

Clock Settings
Other Requirements & Options
Timing Analysis Reporting
New...
Edit...
Delete
Default External Delays...
OK
Cancel

Specifies the default frequency requirement.

☞ You can override the default required f_{MAX} for the project by creating clock settings and assigning them to individual clock signals in your design, as described in "Session 12: Perform Multi-Clock Timing Analysis" on page 79.

2. Cut Timing Paths

You can cut specific signal paths to direct the Timing Analyzer to ignore false paths during timing analysis. You can turn on options in the **Other Requirements & Options** tab of the **Timing Settings** dialog box to cut all instances of specific categories of paths, or you can cut specific paths on a node-by-node basis with the **Cut Timing Path** option in the **Assignment Organizer** dialog box.

To cut off feedback from I/O pins, follow these steps:

1. In the **Timing Settings** dialog box, click the **Other Requirements & Options** tab.

2. Make sure that **Cut off feedback from I/O pins** is turned on.

3. Click **OK**. The Timing Analyzer ignores all signal feedback from within the device.

Go to "Specifying Project-Wide Timing Requirements" in Quartus Help for information about specifying t_{SU}, t_{H}, t_{CO}, and t_{PD} timing requirements.

Session 12: Perform Multi-Clock Timing Analysis

The Quartus Timing Analyzer automatically performs single-clock frequency analysis as part of compilation. However, the Timing Analyzer can also perform timing analysis on designs that use complex timing assignments, such as designs with multiple clocks.

To perform an accurate multi-clock analysis, you must first define the relationship between all clocks in the design. To define this relationship, you must specify which signals function as the absolute clocks (not dependent on other clocks) and the derived clocks (dependent on other clocks) in the design. In the Quartus software, you define the behavior and requirements of clocks by creating clock settings. Then, you assign these settings to the clock signals in the design.

This session includes the following steps:

1. Create clock settings.
2. Assign the clock settings to a pin.
3. Rerun timing analysis.
4. View the Clock Requirement timing analysis section.

1. Create Clock Settings

To create an absolute clock setting, follow these steps:

1. Choose **Timing Settings** (Project menu). The **Clock Settings** tab opens automatically.

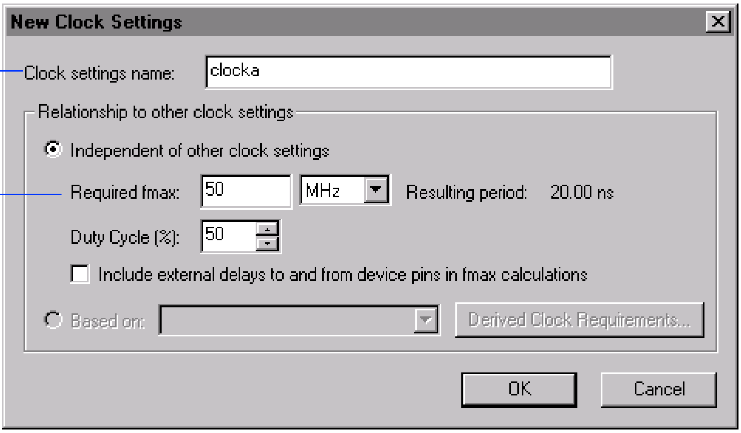
2. Under **Specify circuit frequency as**, select **Settings for individual clock signals**.

3. Click **New**. The **New Clock Settings** dialog box opens.

4. In the **Clock settings name** box, type `clocka` as the name for the new group of clock settings.

5. To specify that these clock settings are for an absolute clock, under **Relationship to other clock settings**, make sure that **Independent of other clock settings** is selected.

6. To specify the required $\mathbf{f}_{MAX}$ of the absolute clock, in the **Required fmax** box, type `50` and select **MHz** in the list. See the following illustration:



Specifies the clock settings name.

Specifies the required frequency of the absolute clock.

New Clock Settings

Clock settings name: clocka

Relationship to other clock settings

● Independent of other clock settings

Required fmax: 50 MHz Resulting period: 20.00 ns

Duty Cycle (%): 50

□ Include external delays to and from device pins in fmax calculations

○ Based on:

7. Click **OK**. The **clocka** settings appear in the **Existing clock settings** list.

To create a derived clock setting, follow these steps:

1. Click **New**. The **New Clock Settings** dialog box opens.

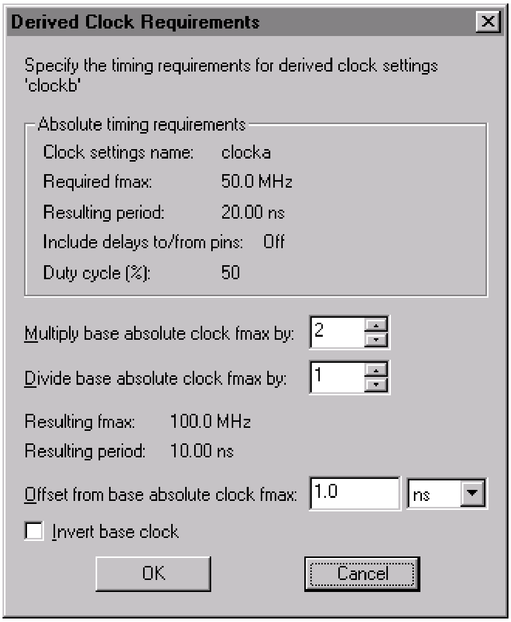
2. In the **Clock settings name** box, type `clockb`.

3. Under **Relationship to other clock settings**, select **Based on** and select **clocka** in the list.

4. To specify timing requirements for the derived clock, click **Derived Clock Requirements**. The **Derived Clock Requirements** dialog box opens.

5. To specify that the derived clock is a multiple of two of the absolute clock, select **2** in the **Multiply base absolute clock fmax by** box.

6. To specify that the derived clock is offset from the absolute clock, type `1.0` and select **ns** in the **Offset from base absolute clock fmax** box. See the following illustration:



Derived Clock Requirements

Specify the timing requirements for derived clock settings 'clockb'

Absolute timing requirements

Clock settings name: clocka

Required fmax: 50.0 MHz

Resulting period: 20.00 ns

Include delays to/from pins: Off

Duty cycle (%): 50

Multiply base absolute clock fmax by: 2

Divide base absolute clock fmax by: 1

Resulting fmax: 100.0 MHz

Resulting period: 10.00 ns

Offset from base absolute clock fmax: 1.0 ns

☐ Invert base clock

7. Click **OK**.

8. In the **New Clock Settings** dialog box, click **OK** to add the **clockb** clock settings to the **Existing clock settings** list.

9. In the **Timing Settings** dialog box, click **OK**.

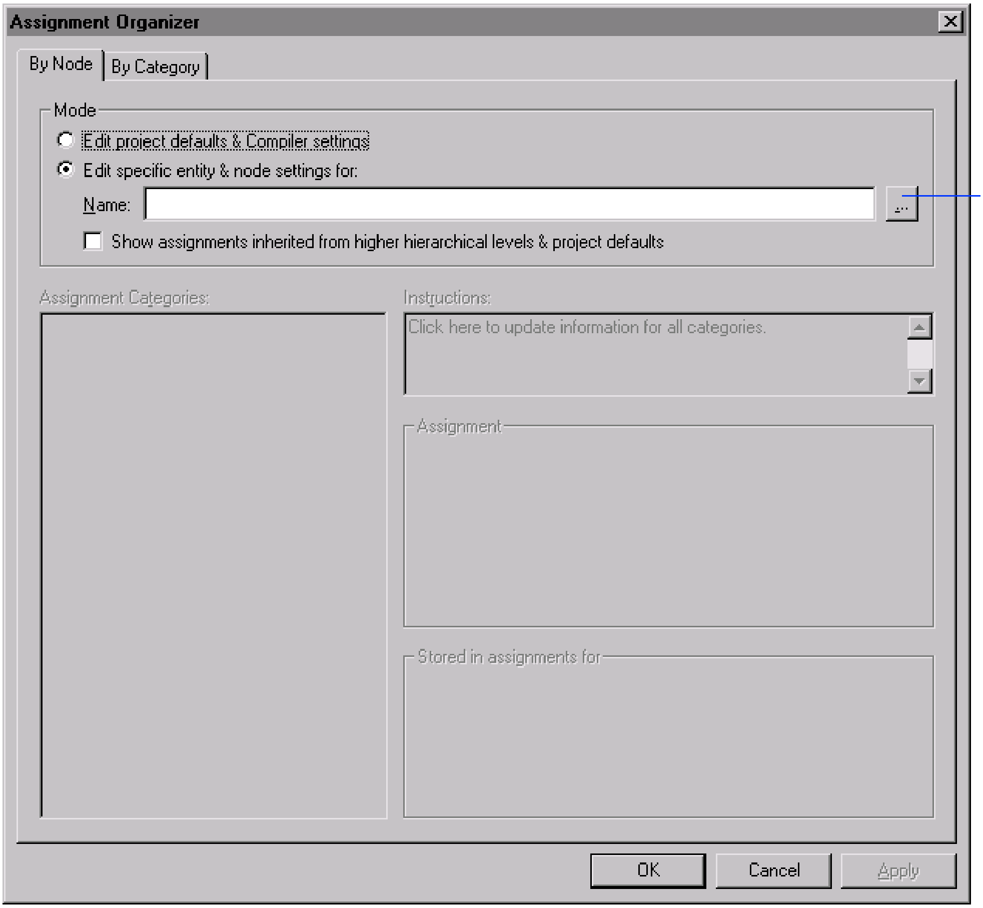
2. Assign the Clock Settings to a Pin

Once you have defined an absolute or derived clock setting, you must assign the setting to the appropriate clock signal(s) in the design. In the Quartus software, you use the **Assignment Organizer** command (Tools menu) to create, delete, edit, and view the assignments in the project.

To assign the **clocka** settings to the `clk` pin, follow these steps:

1. Choose **Assignment Organizer** (Tools menu). The **By Node** tab appears automatically.

2. Under **Mode**, select **Edit specific entity and node settings for**, as shown in the following illustration:



Assignment Organizer

By Node
By Category
Mode
○ Edit project defaults & Compiler settings
● Edit specific entity & node settings for:
Name:
...
□ Show assignments inherited from higher hierarchical levels & project defaults
Assignment Categories:
Instructions:
Click here to update information for all categories.
Assignment
Stored in assignments for
OK
Cancel
Apply

Opens the Node Finder.

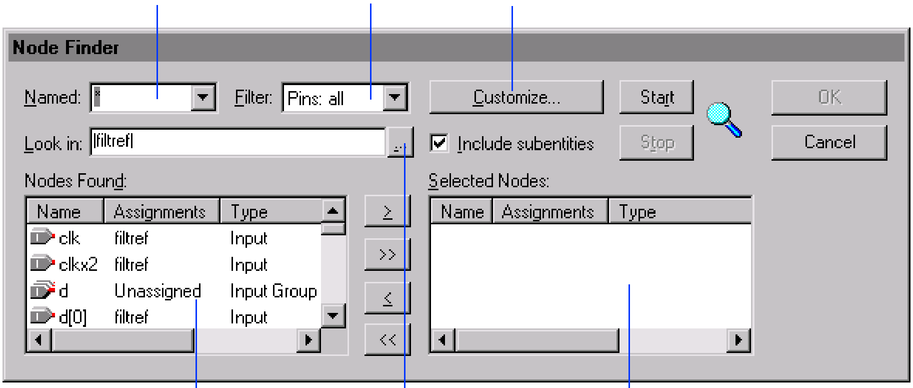
3. Under **Mode**, click **Browse** (**...**) next to the **Name** box. The **Node Finder** dialog box opens.

4. To find the node you want to assign, in the Node Finder, select **Pins: all** in the **Filter** list and click **Start**. See the following illustration:

Specifies a node name search string.

Shows the current filter.

Creates a custom filter.



Name	Assignments	Type
clk	filtref	Input
clkx2	filtref	Input
d	Unassigned	Input Group
d[0]	filtref	Input

Shows all nodes found in the current search.

Specifies the search hierarchy level.

Copies node names from the Node Finder.

5. In the **Nodes Found** list, double-click the `clk` pin name.

6. To copy the `clk` pin name from the **Node Finder** dialog box to the **Assignment Organizer** dialog box, click **OK**. You are returned to the **Assignment Organizer** dialog box.

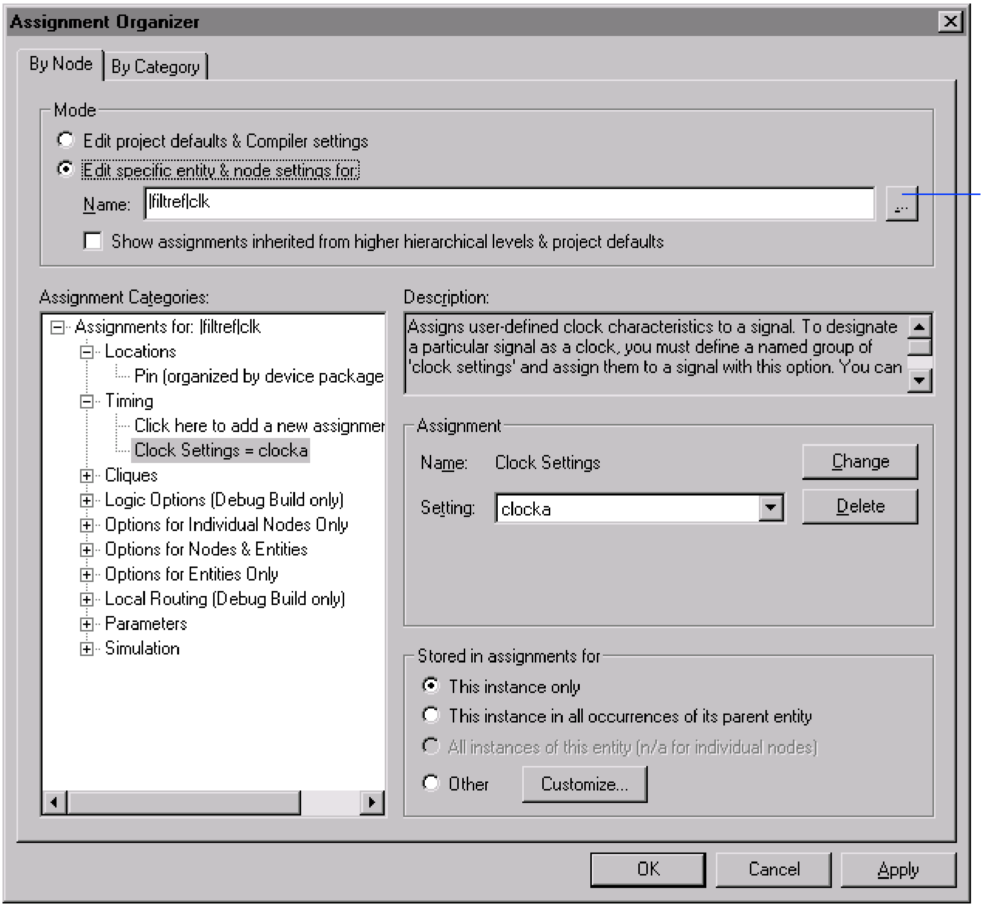
7. In the **Assignment categories** list, click the + icon to expand **Timing**.

8. Click the **Click here to add a new assignment** text.

9. Under **Assignment**, in the **Name** list, make sure that **Clock Settings** is selected.

10. To identify the clock settings that should be assigned to the pin, make sure that **clocka** is selected in the **Settings** list.

11. Click **Add**. The new assignment appears in the **Assignment Categories** list, as shown in the following illustration:



Assignment Organizer

By Node | By Category

Mode

○ Edit project defaults & Compiler settings

● Edit specific entity & node settings for:

Name: |filtref|clk

□ Show assignments inherited from higher hierarchical levels & project defaults

Assignment Categories:

- Assignments for: |filtref|clk
  - Locations
    - Pin (organized by device package
  - Timing
    - Click here to add a new assignmer
    - Clock Settings = clocka
  - Cliques
  - Logic Options (Debug Build only)
  - Options for Individual Nodes Only
  - Options for Nodes & Entities
  - Options for Entities Only
  - Local Routing (Debug Build only)
  - Parameters
  - Simulation

Description:

Assigns user-defined clock characteristics to a signal. To designate a particular signal as a clock, you must define a named group of 'clock settings' and assign them to a signal with this option. You can

Assignment

Name: Clock Settings

Setting: clocka

Stored in assignments for

● This instance only

○ This instance in all occurrences of its parent entity

○ All instances of this entity (n/a for individual nodes)

○ Other

Opens the Node Finder.

12. Click **OK**.

13. Repeat steps 1 through 12 to assign the **clockb** settings to the `clkx2` pin.

You can also specify many other types of individual timing assignments. Go to “Making Individual Timing Assignments” in Quartus Help for more information on timing assignments.

3. Rerun Timing Analysis

To rerun timing analysis:

1. Choose **Start Timing Analysis** (Processing menu).

2. When a dialog box prompt asks whether you want to recompile before starting timing analysis, click **No**. The timing analysis runs without recompiling the design.

   ☞ The Quartus software asks you if you want to recompile at this point because specifying timing requirements affects fitting when you are using timing-driven compilation. For this tutorial, recompilation is not necessary.

3. When a dialog box prompt informs you that the timing analysis was unsuccessful, click **OK**. Timing analysis was unsuccessful because the specified timing requirements were not achieved. The following sections explain how to make this timing analysis successful.

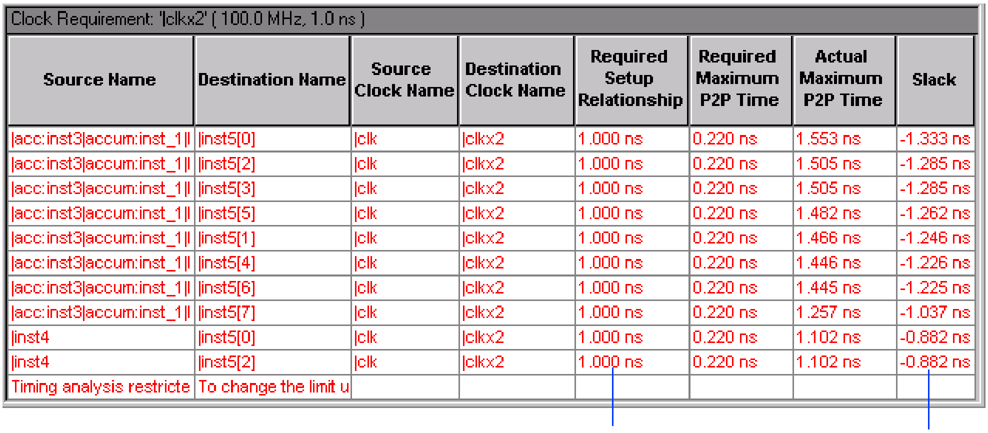
4. View the Clock Requirement Timing Analysis Section

In designs that use complex timing assignments, the Timing Analyzer generates a Clock Requirement section for each clock signal. The Clock Requirement section reports the “slack” times of the timing requirement paths. Slack is the margin by which a timing requirement was achieved or not achieved. A positive slack, displayed in black, indicates that the requirement was achieved. A negative slack, displayed in red, indicates that the requirement was not achieved.

To view the Clock Requirement section for the `clkx2` signal, follow these steps:

1. In the left pane of the Compilation Report window, click the + icon to expand the **Timing Analyses** folder.

2. Under the **Timing Analyses** folder, select the Clock Requirement section for the `clkx2` signal. The timing analysis information is displayed in a multi-column table. If necessary, drag the resizing tool to resize columns in the table. See the following illustration:



Clock Requirement: '|clkx2' (100.0 MHz, 1.0 ns)

Source Name	Destination Name	Source Clock Name	Destination Clock Name	Required Setup Relationship	Required Maximum P2P Time	Actual Maximum P2P Time	Slack
|acc:inst3|accum:inst_1|l	|inst5[0]	|clk	|clkx2	1.000 ns	0.220 ns	1.553 ns	-1.333 ns
|acc:inst3|accum:inst_1|l	|inst5[2]	|clk	|clkx2	1.000 ns	0.220 ns	1.505 ns	-1.285 ns
|acc:inst3|accum:inst_1|l	|inst5[3]	|clk	|clkx2	1.000 ns	0.220 ns	1.505 ns	-1.285 ns
|acc:inst3|accum:inst_1|l	|inst5[5]	|clk	|clkx2	1.000 ns	0.220 ns	1.482 ns	-1.262 ns
|acc:inst3|accum:inst_1|l	|inst5[1]	|clk	|clkx2	1.000 ns	0.220 ns	1.466 ns	-1.246 ns
|acc:inst3|accum:inst_1|l	|inst5[4]	|clk	|clkx2	1.000 ns	0.220 ns	1.446 ns	-1.226 ns
|acc:inst3|accum:inst_1|l	|inst5[6]	|clk	|clkx2	1.000 ns	0.220 ns	1.445 ns	-1.225 ns
|acc:inst3|accum:inst_1|l	|inst5[7]	|clk	|clkx2	1.000 ns	0.220 ns	1.257 ns	-1.037 ns
|inst4	|inst5[0]	|clk	|clkx2	1.000 ns	0.220 ns	1.102 ns	-0.882 ns
|inst4	|inst5[2]	|clk	|clkx2	1.000 ns	0.220 ns	1.102 ns	-0.882 ns
Timing analysis restricte	To change the limit u						

Minimum setup requirement imposed by 1.0 ns offset of clockb settings.

Slack times in red indicate unachieved requirements.

The actual timing values may differ from those in the previous illustration due to improvements made in the Quartus software after this manual was printed.

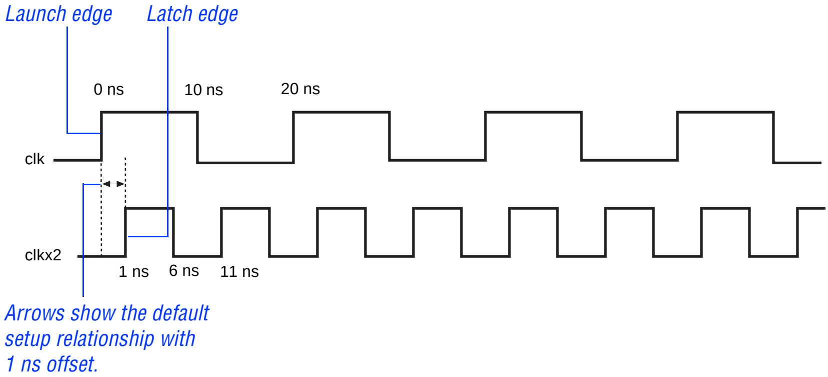
The Clock Requirement section for the `clkx2` signal displays slack times in red, indicating that the specified f_{MAX} requirement was not achieved along the path. “Session 13: Specify a Multicycle Path,” next, describes how to specify a multicycle path that corrects this condition.

Session 13: Specify a Multicycle Path

In the **fir_filter** design, the 1.0 ns offset you specified for the **clockb** clock settings causes the Timing Analyzer to report that timing requirements were not met. This condition occurs because, by default, the Timing Analyzer uses the most restrictive setup relationship when analyzing paths between registers that are clocked by different clocks. The Timing Analyzer verifies that the data is present at the destination register in time to be properly latched.

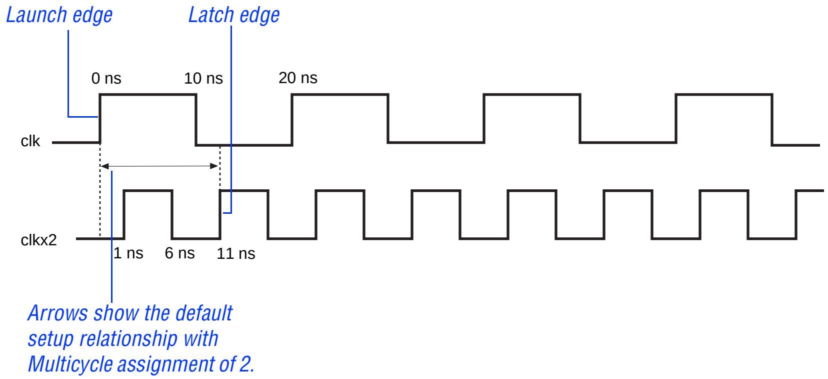
In designs with multiple clocks, for every latch edge on the destination clock, the launching edge on the source register determines the delay requirement for the path. In the **fir_filter** design, for every latch edge of the destination register, the Timing Analyzer finds the nearest launch edge that precedes each capture edge. The smallest difference determines the maximum delay requirement. Figure 5 shows this condition in the **fir_filter** design:

Figure 5. Default Multi-Clock Setup Relationship


Launch edge
Latch edge
0 ns
10 ns
20 ns
clk
clkx2
1 ns
6 ns
11 ns
Arrows show the default setup relationship with 1 ns offset.

You can use the Multicycle timing assignment to specify a path that requires more than one clock cycle to propagate. Assigning a Multicycle of 2 to all registers clocked by `clkx2` allows you to override the default setup relationship and delay the latch edge by one clock cycle. Figure 6 shows the multi-clock setup relationship after creating a Multicycle assignment of 2.

Figure 6. Multi-Clock Setup Relationship with Multicycle Assignment of 2


Launch edge
Latch edge
0 ns
10 ns
20 ns
clk
clkx2
1 ns
6 ns
11 ns
Arrows show the default setup relationship with Multicycle assignment of 2.

This session includes the following steps:

1. Create a Multicycle timing assignment.
2. Rerun timing analysis.
3. View the Clock Requirement timing analysis section.

1. Create a Multicycle Timing Assignment

When you make a point-to-point assignment between two clock signals, the assignment is automatically applied to all register-to-register paths between the two clocks. To add the Multicycle assignment to all register-to-register paths between the `clk` and `clkx2` pins, follow these steps:

1. If necessary, to open the **filtref.bdf** block diagram, choose **Open** (File menu) and select **filtref.bdf**.

2. In the **filtref.bdf** block diagram, select the `clkx2` input pin.

3. Choose **Assignment Organizer** (right button pop-up menu). The **Assignment Organizer** dialog box opens, with the **Edit specific entity & node settings for** option selected, and with the hierarchical path name of the `clkx2` pin shown in the **Name** box.

4. In the **Assignment Categories** list, click the + icon to expand **Timing**.

5. Click the **Click here to add new assignment** text.

6. Under **Assignment**, select **Multicycle** in the **Name** list.

7. To specify a multicycle path that requires two clock cycles to propagate, in the **Setting** box, type `2`.

8. To specify a point-to-point assignment, in the **Fed by** box, type `clk` or click **Browse (...)** to select it with the Node Finder.

9. Click **Add**. The assignment then appears in the **Assignment Categories** list.

10. Click **OK**.

2. Rerun Timing Analysis

To rerun timing analysis:

1. Choose **Start Timing Analysis** (Processing menu).

2. When a dialog box prompt asks whether you want to recompile before starting timing analysis, click **No**. The timing analysis runs without recompiling the design.

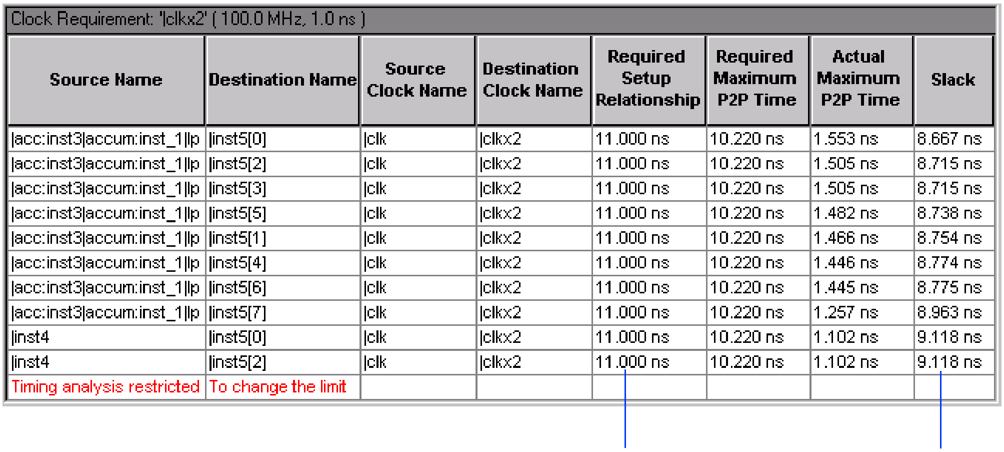
3. When a dialog box prompt informs you that the timing analysis was successful, click **OK**. Timing analysis was successful because the specified timing requirements were achieved.

3. View the Clock Requirement Timing Analysis Section

To view the timing analysis results to determine how setting the Multicycle assignment affected timing requirements, follow these steps:

1. In the left pane of the Compilation Report window, click the + icon to expand the **Timing Analyses** folder.

2. Under the **Timing Analyses** folder, select the Clock Requirement section for the `clkx2` signal. The Clock Requirement section displays all slack times in black, indicating that all timing requirements were achieved after overriding the default setup relationship with the Multicycle assignment. See the following illustration:



Clock Requirement: '|clkx2' (100.0 MHz, 1.0 ns)

Source Name	Destination Name	Source Clock Name	Destination Clock Name	Required Setup Relationship	Required Maximum P2P Time	Actual Maximum P2P Time	Slack
|acc:inst3|accum:inst_1|lp	|inst5[0]	|clk	|clkx2	11.000 ns	10.220 ns	1.553 ns	8.667 ns
|acc:inst3|accum:inst_1|lp	|inst5[2]	|clk	|clkx2	11.000 ns	10.220 ns	1.505 ns	8.715 ns
|acc:inst3|accum:inst_1|lp	|inst5[3]	|clk	|clkx2	11.000 ns	10.220 ns	1.505 ns	8.715 ns
|acc:inst3|accum:inst_1|lp	|inst5[5]	|clk	|clkx2	11.000 ns	10.220 ns	1.482 ns	8.738 ns
|acc:inst3|accum:inst_1|lp	|inst5[1]	|clk	|clkx2	11.000 ns	10.220 ns	1.466 ns	8.754 ns
|acc:inst3|accum:inst_1|lp	|inst5[4]	|clk	|clkx2	11.000 ns	10.220 ns	1.446 ns	8.774 ns
|acc:inst3|accum:inst_1|lp	|inst5[6]	|clk	|clkx2	11.000 ns	10.220 ns	1.445 ns	8.775 ns
|acc:inst3|accum:inst_1|lp	|inst5[7]	|clk	|clkx2	11.000 ns	10.220 ns	1.257 ns	8.963 ns
|inst4	|inst5[0]	|clk	|clkx2	11.000 ns	10.220 ns	1.102 ns	9.118 ns
|inst4	|inst5[2]	|clk	|clkx2	11.000 ns	10.220 ns	1.102 ns	9.118 ns
Timing analysis restricted	To change the limit						

Minimum setup requirement reflects the Multicycle assignment of 2.

Slack times in black indicate achieved requirements.

The actual timing values may differ from those in the previous illustration due to improvements made in the Quartus software after this manual was printed.

Simulation

Simulation allows you to test a design thoroughly to ensure that it responds correctly in every possible situation before you program or configure it into a device.

You must supply input vectors as the stimuli for the Quartus Simulator. The Simulator uses these input vectors to simulate the output signals that a programmed device would produce under the same conditions. In a typical simulation session, you create multiple sets of input vectors and check the resulting outputs.

Depending on the type of information you need, you can perform functional or timing simulation with the Simulator. Functional simulation tests only the logical operation of a design, while timing simulation tests both the logical operation and the worst-case timing for the design in the target device.

The following sessions guide you through the steps necessary to create a Vector Waveform File (**.vwf**), specify Simulator settings, run a timing simulation, and analyze the simulation results.

Session 14: Create a Waveform File for Simulation

You can create Vector Waveform Files (**.vwf**) with the Quartus Waveform Editor to use as stimuli for simulation. VWFs describe the simulation input vectors and simulation outputs as graphical waveforms. You can also specify vector stimuli in a text-based Vector File (**.vec**). The Quartus software allows you to open vector files either in text or waveform format.

☞ If you are already familiar with creating VWFs with the Quartus Waveform Editor, you can copy the Altera-provided **fir.vwf** file from the **\qdesigns\tutorial** subdirectory into the **\qdesigns\fir_filter** subdirectory. Altera recommends that you copy tutorial files by opening them in the Quartus software with the **Open** command (File menu), choosing **Save As** (File menu), turning on

Add file to current project, and then saving the file to the **\qdesigns\fir_filter** subdirectory. If you copy the Altera-provided file, skip to "Session 15: Specify Simulator Settings" on page 98.

This session includes the following steps:

1. Create a new VWF.
2. Add input and output nodes to the file.
3. Edit the input node waveforms.

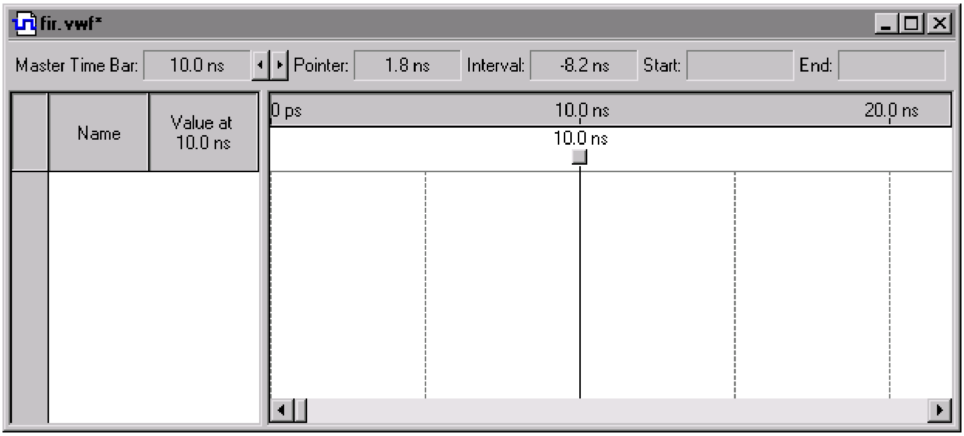
1. Create a New Vector Waveform File

To create a VWF, follow these steps:

1. Choose **New** (File menu).

2. To select VWF as the file type, click the **Other Files** tab and select **Vector Waveform File**.

3. Click **OK**. The Waveform Editor opens, displaying an empty waveform file, as shown in the following illustration:



4. To change the end time for the file, choose **End Time** (Time menu).

5. In the **Time** box, type `700` and select **ns** in the list.

6. Click **OK**.

7. To save the file as **fir.vwf**, choose **Save As** (File menu), type `fir` in the **File name** box, and click **Save**.

2. Add Input & Output Nodes to the File

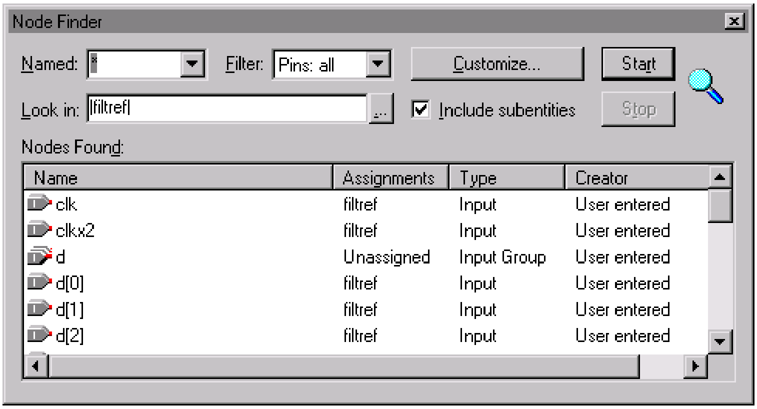
You can create a VWF by entering some or all of the input node waveforms and the desired output node waveforms.

To add the nodes, follow these steps:

1. To find the node names you want to add to the file, turn on **Auxiliary Windows > Node Finder** (View menu). The **Node Finder** dialog box is displayed.

2. In the Node Finder, select **Pins: all** in the **Filter** list.

3. To find the nodes you want to add to the VWF, click **Start**. See the following illustration:



Named: *

Filter: Pins: all

Look in: |filtref|

☑ Include subentities

Nodes Found:

Name	Assignments	Type	Creator
clk	filtref	Input	User entered
clkx2	filtref	Input	User entered
d	Unassigned	Input Group	User entered
d[0]	filtref	Input	User entered
d[1]	filtref	Input	User entered
d[2]	filtref	Input	User entered

4. In the **Nodes Found** list, select the `clk`, `clkx2`, `d`, `newt`, `reset`, `yvalid`, `next`, and `yn_out` pins and drag them into the **Name** column of the VWF. You can select multiple contiguous names with Shift+Click or select multiple non-contiguous names with Ctrl+Click.

5. To close the Node Finder, turn off **Auxiliary Windows > Node Finder** (View menu).

You can also create custom filters for finding nodes in the design with the Node Finder. Go to “Creating a Custom Filter” in Quartus Help for specific instructions about creating a custom filter.

3. Edit the Input Node Waveforms

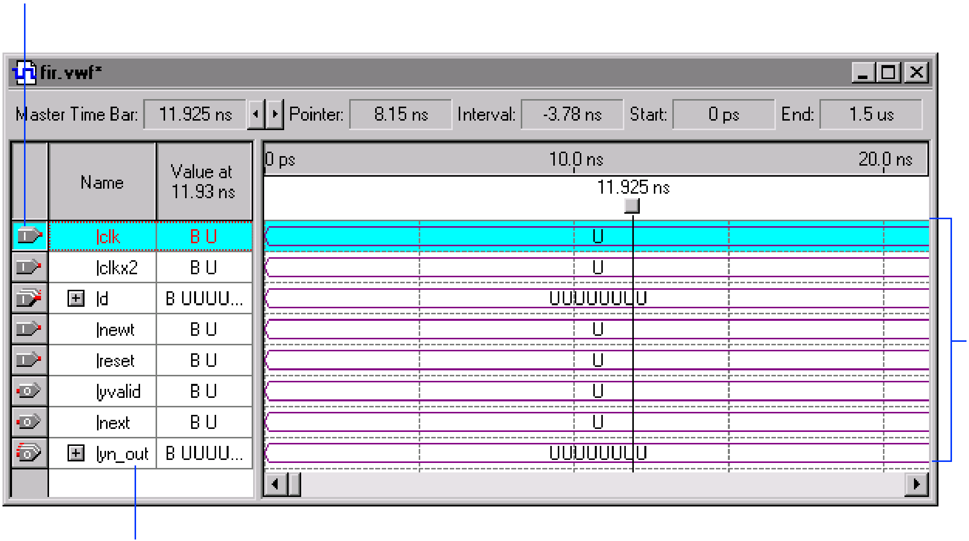
You create the input vectors for simulation by specifying the logic levels of the node waveforms. The logic level changes represent the behavior of signals in the design.

To edit the `clk` input node waveform, follow these steps:

1. If necessary, click the **Selection Tool** button on the Waveform Editor toolbar.

2. To select the entire `clk` input node waveform, click the Selection Tool on the “handle” of the `clk` node. The entire node is highlighted. See the following illustration:

Node handle



fir.vwf*

Master Time Bar: 11.925 ns Pointer: 8.15 ns Interval: -3.78 ns Start: 0 ps End: 1.5 us

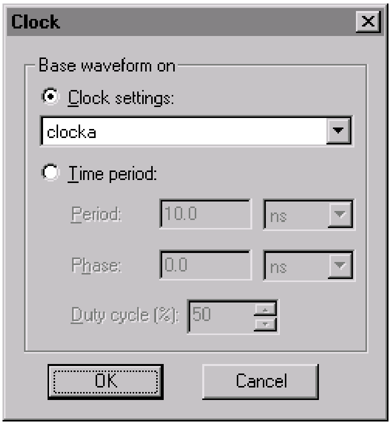
Name	Value at 11.93 ns
clk	B U
clkx2	B U
d	B UUUU...
newt	B U
reset	B U
yvalid	B U
next	B U
yn_out	B UUUU...

Waveform display

Lists input and output node and bus names.

3. Choose **Clock** (Value menu).

4. Under **Base waveform on**, select **Clock settings** and select **clocka** in the list, as shown in the following illustration:



Clock

Base waveform on

● Clock settings:

clocka

○ Time period:

Period: 10.0 ns

Phase: 0.0 ns

Duty cycle (%): 50

5. Click **OK**. The `clk` waveform is created according to the **clocka** clock settings.

To edit the `clkx2` input node waveform, follow these steps:

1. Click the **Selection Tool** button on the Waveform Editor toolbar.

2. To select the entire `clkx2` input node waveform, click the Selection Tool on the “handle” of the `clkx2` node. The entire node is highlighted.

3. Choose **Clock** (Value menu).

4. Under **Base waveform on**, select **Clock settings** and select **clockb** in the list.

5. Click **OK**. The `clkx2` waveform is created according to the **clockb** clock settings.

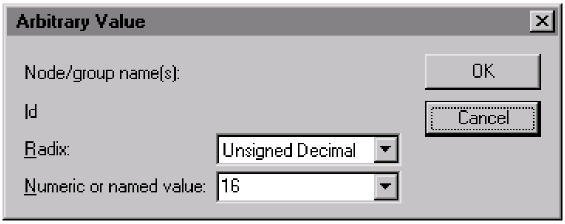
To edit the `d` input bus waveform, follow these steps:

1. To select the entire `d` input bus waveform, click the Selection Tool on the waveform’s handle. The entire bus, including the waveform, is highlighted.

2. Choose **Arbitrary Value** (Value menu).

3. In the **Radix** list, select **Unsigned Decimal**.

4. In the **Numeric or named value** list, type 16, as shown in the following illustration:



Arbitrary Value

Node/group name(s):

ld

Radix: Unsigned Decimal

Numeric or named value: 16

5. Click **OK**. The d input bus waveform is displayed with a binary value of 16.

To edit the newt input node waveform, follow these steps:

1. To select the entire newt input node, click the Selection Tool on the node's handle. The entire node is highlighted.

2. Choose **Clock** (Value menu).

3. Under **Base waveform on**, select **Time period**.

4. Type 80 in the **Period** box, and select **ns** in the list.

5. In the **Duty Cycle** list, specify 25.

6. Click **OK**. The waveform displays a repeating waveform with a period of 80 ns and a duty cycle of 25% on the newt input signal.

To edit the reset input node waveform, follow these steps:

1. Click the Selection Tool at time 0 ns on the reset input waveform and drag the pointer to time 20 ns.

2. Press the right mouse button on the selected interval and choose **Value** > **Forcing Low** (right button pop-up menu).

3. Click the Selection Tool at time 20 ns on the reset input waveform and drag the pointer to time 40 ns.

4. Press the right mouse button on the selected interval and choose **Value > Forcing High** (right button pop-up menu).

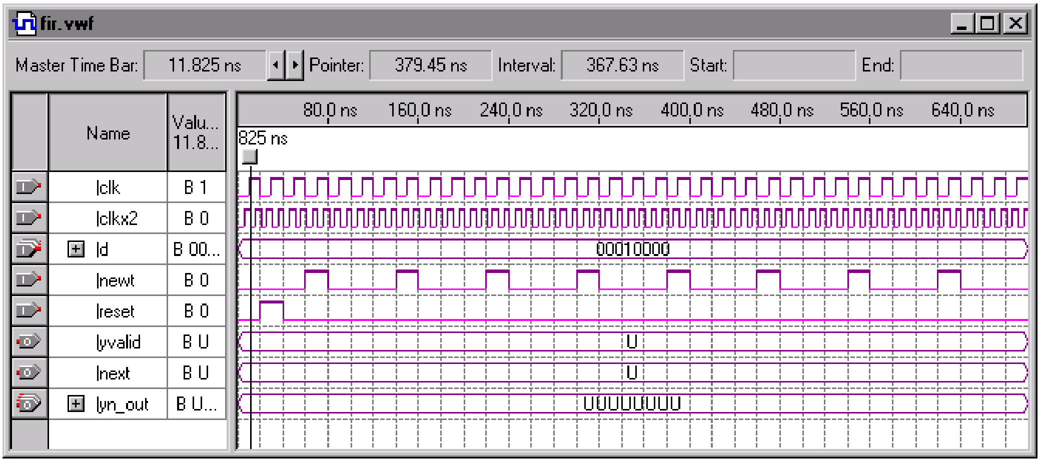
5. To see the entire waveform, choose **Fit in Window** (View menu).

6. Click the Selection Tool at time 40 ns on the `reset` input waveform and drag the pointer to the end of the stimulus file.

7. Press the right mouse button on the selected interval and choose **Value > Forcing Low** (right button pop-up menu).

8. To save the file, choose **Save** (File menu).

The waveforms appear as shown in the following illustration:



fir.vwf

Master Time Bar: 11.825 ns Pointer: 379.45 ns Interval: 367.63 ns Start: End:

Name	Valu... 11.8...
clk	B 1
clkx2	B 0
ld	B 00...
newt	B 0
reset	B 0
yvalid	B U
next	B U
yn_out	B U...

Session 15: Specify Simulator Settings

The Quartus software allows you to simulate an entire design, or to simulate any part of a design. You can designate any top-level design entity in a project as the “simulation focus,” which is the design entity you want to simulate.

The Simulator settings allow you to specify the simulation focus, the type of simulation that should be performed, the time period covered by the simulation, the source of vector stimuli, and other options.

You can create and save your own customized groups of settings for use with the Quartus Simulator, or you can use the settings that are generated automatically each time you create a new project.

This session includes the following steps:

1. View the Simulator general settings.
2. Specify Simulator time and vectors settings.
3. Specify Simulator mode settings.
4. Specify Simulator options.

☞ The procedures below explain how to view and edit Simulator settings using menu commands and dialog boxes. However, you can also easily specify all Simulator settings by following the steps in the **Simulator Settings Wizard** (Processing menu).

1. View the Simulator General Settings

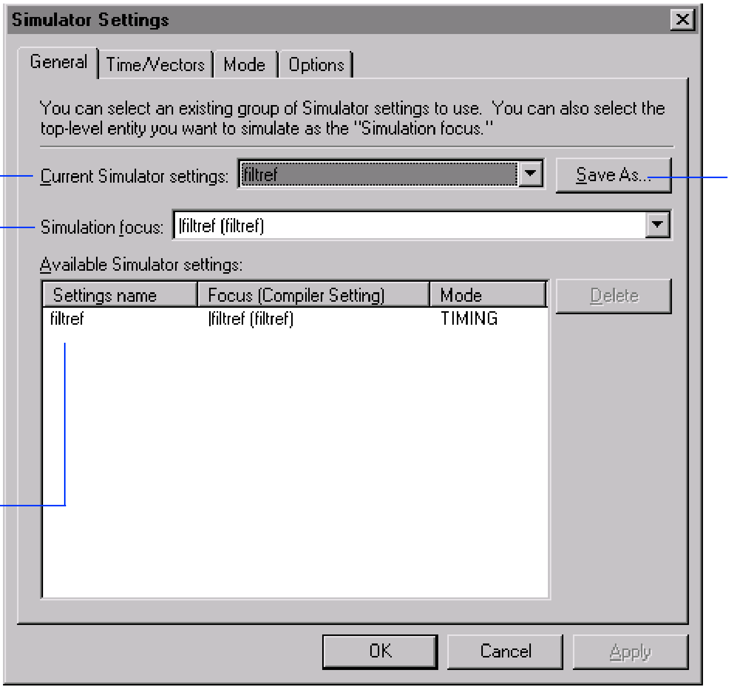
The **General** tab of the **Simulator Settings** dialog box (Processing menu) allows you to select an existing group of Simulator settings, define and save a new group of Simulator settings, specify the simulation focus, or delete existing settings.

To view the default Simulator general settings created for the current project, follow these steps:

1. Make sure that you are in Simulation mode by selecting **Simulate Mode** (Processing menu).

2. Choose **Simulator Settings** (Processing menu). The **General** tab opens automatically.

   The **General** tab displays the default Simulator general settings created by the Quartus software when you created the project, as shown in the following illustration:



You can select an existing group of Simulator settings to use. You can also select the top-level entity you want to simulate as the "Simulation focus."

Current Simulator settings: filtref

Simulation focus: |filtref (filtref)

Available Simulator settings:

Settings name	Focus (Compiler Setting)	Mode
filtref	|filtref (filtref)	TIMING

Specifies the current Simulator settings name.

Creates a new group of Simulator settings.

Specifies the hierarchical path name of the design entity you want to simulate, and the associated Compiler settings name.

Shows the existing Simulator settings in your project.

2. Specify Simulator Time & Vectors Settings

The **Time/Vectors** tab of the **Simulator Settings** dialog box allows you to specify the time period covered by the simulation, and the source of vector stimuli. You can provide vector stimuli for simulation in a Vector Waveform File (**.vwf**) or a text-based Vector File (**.vec**), or you can enter vector stimuli in the Tcl Console window.

To specify the simulation time period and the source of vector stimuli, follow these steps:

1. In the **Simulator Settings** dialog box, click the **Time/Vectors** tab.

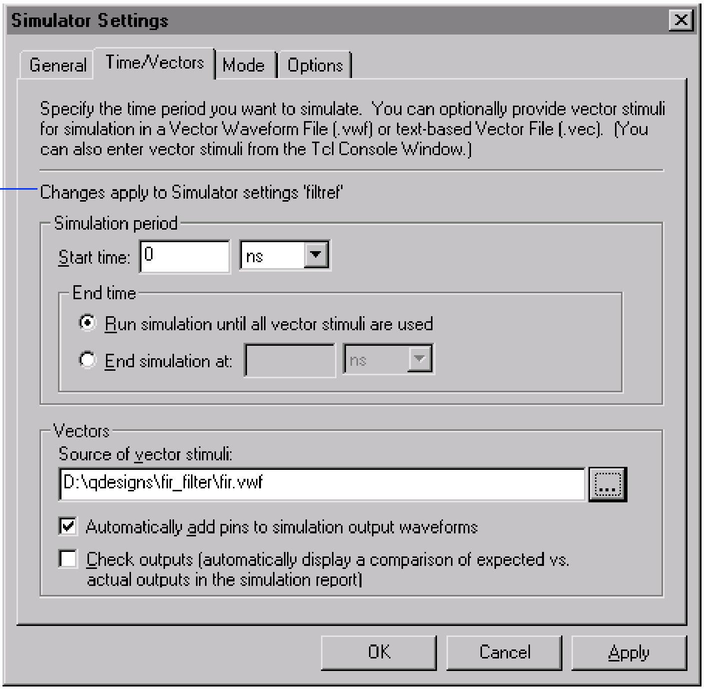
2. Under **Simulation period**, in the **Start time** box, type `0` and select **ns** in the list.

3. Under **End time**, make sure that **Run simulation until all vector stimuli are used** is selected.

4. Under **Vectors**, in the **Source of vector stimuli** box, type `D:\qdesigns\fir_filter\fir.vwf`, or click **Browse** (**...**) to select the file.

> ☞ When you run a simulation, the Simulator automatically searches for a vector file (**.vwf**, **.vec**, or **.tbl**) with the same name as the current Simulator settings. Therefore, it is not necessary to explicitly specify a vector file if the vector file has the same name as the current Simulator settings.

5. Under **Vectors**, make sure that **Automatically add pins to simulation output waveforms** is turned on. See the following illustration:



Identifies the Simulator settings you are editing.

Simulator Settings

General | Time/Vectors | Mode | Options

Specify the time period you want to simulate. You can optionally provide vector stimuli for simulation in a Vector Waveform File (.vwf) or text-based Vector File (.vec). (You can also enter vector stimuli from the Tcl Console Window.)

Changes apply to Simulator settings 'filtref'

Simulation period

Start time: 0 ns

End time

● Run simulation until all vector stimuli are used

○ End simulation at: ns

Vectors

Source of vector stimuli: D:\qdesigns\fir_filter\fir.vwf [...]

☑ Automatically add pins to simulation output waveforms

□ Check outputs (automatically display a comparison of expected vs. actual outputs in the simulation report)

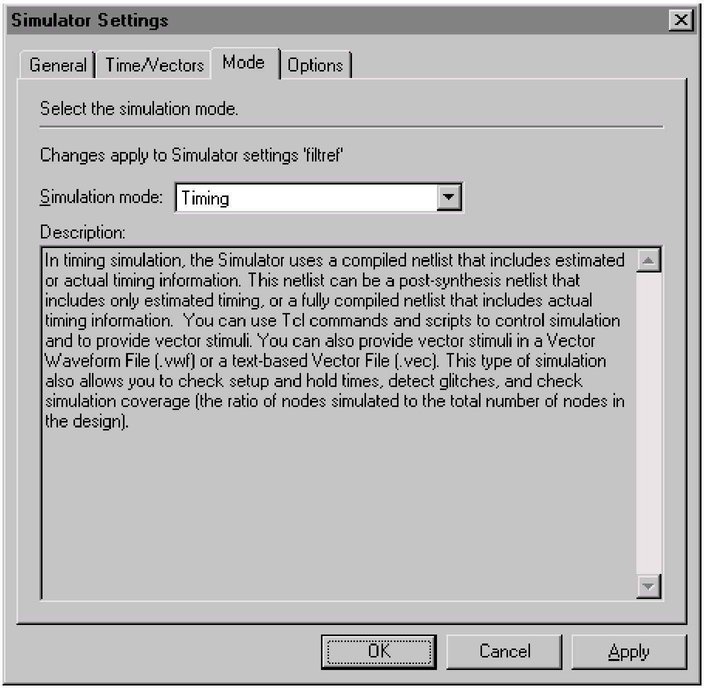
3. Specify Simulator Mode Settings

The **Mode** tab of the **Simulator Settings** dialog box allows you to specify which type of simulation should be performed. Two types of simulation are available: functional simulation and timing simulation. Functional simulation tests the logical operation of the design by simulating the behavior of flattened netlists extracted from the design files. Timing simulation uses a fully compiled netlist, which includes actual timing information, to test the logical and timing performance of the design.

To specify the timing simulation mode, follow these steps:

1. In the **Simulator Settings** dialog box, click the **Mode** tab.

2. In the **Simulation mode** list, select **Timing**. A description of timing simulation appears in the **Description** field, as shown in the following illustration:



Select the simulation mode.

Changes apply to Simulator settings 'filtref'

Simulation mode: Timing

Description:

In timing simulation, the Simulator uses a compiled netlist that includes estimated or actual timing information. This netlist can be a post-synthesis netlist that includes only estimated timing, or a fully compiled netlist that includes actual timing information. You can use Tcl commands and scripts to control simulation and to provide vector stimuli. You can also provide vector stimuli in a Vector Waveform File (.vwf) or a text-based Vector File (.vec). This type of simulation also allows you to check setup and hold times, detect glitches, and check simulation coverage (the ratio of nodes simulated to the total number of nodes in the design).

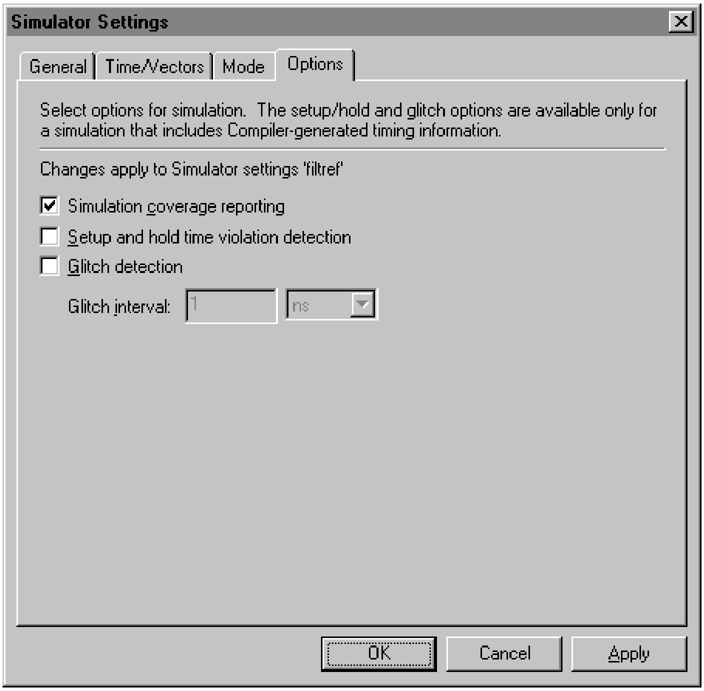
4. Specify Simulator Options

The **Options** tab of the **Simulator Settings** dialog box allows you to specify options that monitor conditions found during simulation.

To monitor the simulation coverage, follow these steps:

1. In the **Simulator Settings** dialog box, click the **Options** tab.

2. Turn on **Simulation coverage reporting**. See the following illustration:



Select options for simulation. The setup/hold and glitch options are available only for a simulation that includes Compiler-generated timing information.

Changes apply to Simulator settings 'filtref'

☑ Simulation coverage reporting

□ Setup and hold time violation detection

□ Glitch detection

Glitch interval: 1 ns

3. Click **OK**. All of the settings and options you specified are saved as the **filtref** Simulator settings. When you run the Simulator, these current Simulator settings are applied to the simulation.

Session 16: Simulate the Design

To run the simulation, follow these steps:

1. Choose **Run Simulation** (Processing menu).

   The Simulator immediately begins to simulate the **filtref** top-level design entity and all of its subordinate design entities, using the **filtref** Simulator settings and the **fir.vwf** vector file. As the Simulator processes the input vectors and simulates the design, the Status window opens automatically, displaying the simulation progress. In addition, the results of the simulation are updated in the Simulation Report window.

   The Simulator runs in the background, freeing your computer for other work. However, the simulation of this design entity is short and you will not have to wait long for it to finish. When you analyze a larger, more complex design, you can always start a simulation and then switch to another application to continue your work.

2. When you receive a message indicating that simulation was successful, click **OK**.

Go to “Creating a Breakpoint” in Quartus Help for more information about breakpoints. Go to “Updating Embedded Memory” in Quartus Help for more information about updating embedded memory.

Session 17: Analyze the Simulation Results

During simulation, the Simulation Report window appears automatically. The Simulation Report displays useful information about the current simulation. By default, the Simulation Waveform section appears in the Simulation Report window. The Simulation Waveform section shows the behavior of the specified outputs during simulation.

The Simulation Report also contains other useful sections that provide information about the current simulation, including information about the Simulator settings, simulation messages, and simulation processing times.

This session includes the following steps:

1. View the Simulation Waveform section.
2. Create a Time Bar.

1. View the Simulation Waveform Section

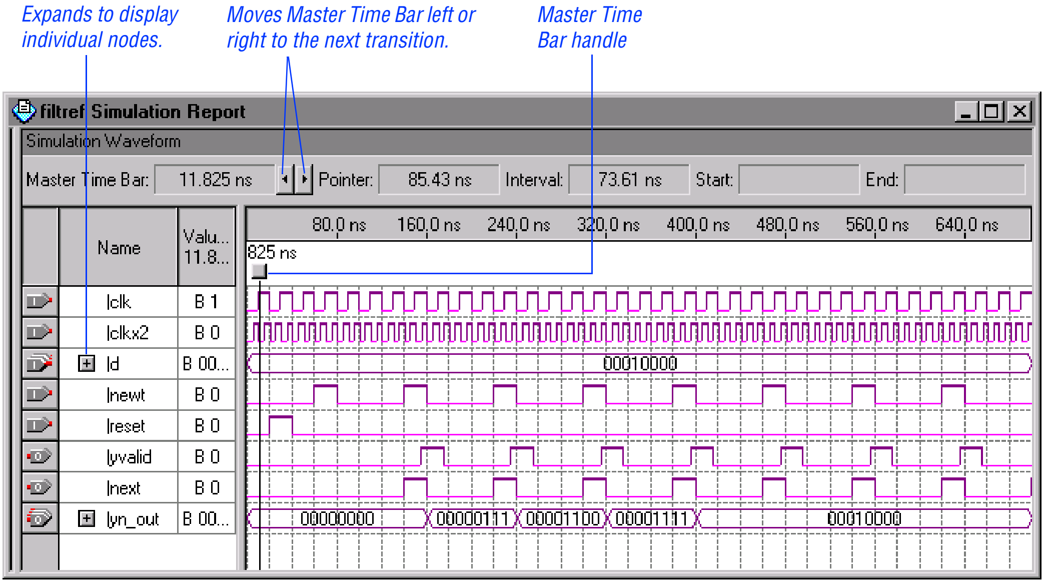
Now that you have simulated the logic levels of the focus entity, the output node waveforms are defined. You can view and analyze the output logic levels in the Simulation Waveform section.

To view the Simulation Waveform section:

1. If necessary, in the left pane of the Simulation Report window, select Simulation Waveform.

2. To see the entire waveform, choose **Fit in Window** (View menu). The simulation waveform appears in the right pane of the Report window, as shown in the following illustration:

Figure 7. Simulation Waveform Section



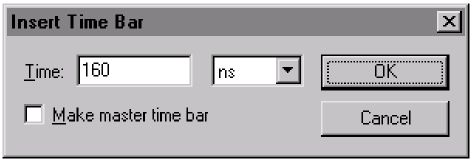
The Waveform Editor allows you to manipulate the data in the following ways:

- You can change the radix used to display group values. Select a group, choose **Properties** (right button pop-up menu), and select a new radix in the **Radix** list.
- You can move the Master Time Bar right and left to successive transitions. Click the movement arrows as shown in Figure 7 on page 105. You can also drag the Master Time Bar by its handle.
- You can zoom in and out with the Zoom Mode Tool on the Waveform Editor toolbar, and with the **Zoom In**, **Zoom Out**, **Fit in Window**, and **Zoom** commands (View menu).
- You can create additional time bars and use them to measure distances between transitions, as described in "2. Create a Time Bar," next.

2. Create a Time Bar

To create a time bar, follow these steps:

1. With the Selection Tool, click the Master Time Bar handle and drag it until the time bar is aligned with the second rising edge of the `clk` signal. At the top of the window, the **Master Time Bar** box displays 30.0 ns.
2. Move the Selection Tool pointer in the waveform to time of 160 ns. At the top of the window, the **Pointer** box displays 160.0 ns, the time where the pointer is located. Note that the **Interval** box displays 130.0 ns, the time difference between the Master Time Bar and the Selection Tool location.
3. Press the right mouse button in the waveform drawing area and choose **Insert Time Bar** (right button pop-up menu).
4. In the **Time** box, type `160` and select **ns** in the list, as shown in the following illustration:



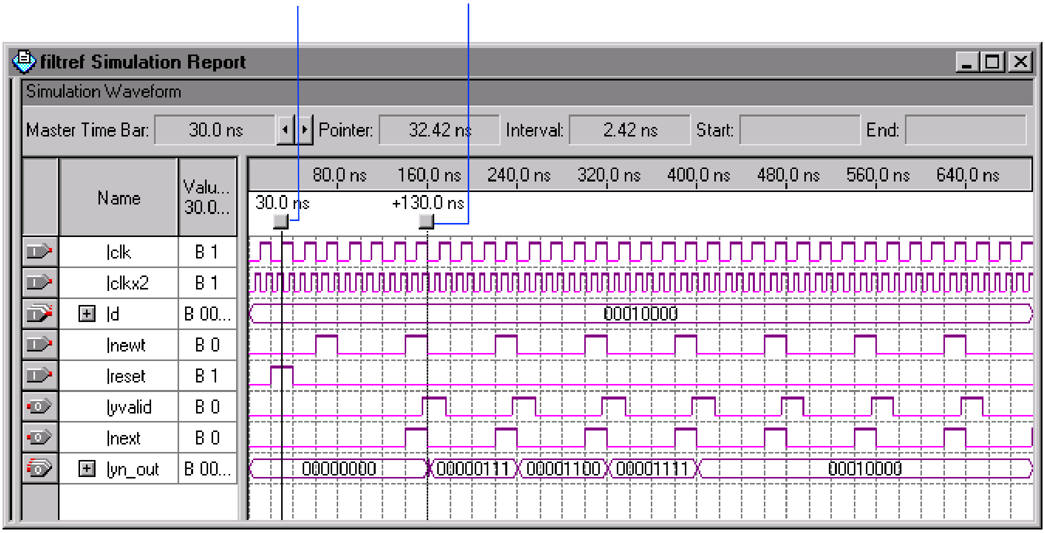
Time: 160 ns

□ Make master time bar

5. Click **OK**. A new time bar appears on the waveform at 160.0 ns. The text +130.0 ns is shown above the second time bar, indicating the time difference from the new time bar to the Master Time Bar. See the following illustration:

Master Time Bar handle

New time bar handle



Master Time Bar: 30.0 ns Pointer: 32.42 ns Interval: 2.42 ns Start: End:

Name	Valu... 30.0...
lclk	B 1
lclkx2	B 1
ld	B 00...
lnewt	B 0
lreset	B 1
lyvalid	B 0
lnext	B 0
lyn_out	B 00...

You can move a time bar by dragging its handle to the desired point in the waveform. All measurement data is updated accordingly.

Programming

With the Quartus Programmer, you can configure Altera APEX 20K devices and program the EPC2 configuration device. You can also verify, examine, and blank-check EPC2 configuration devices. The Programmer and programming hardware (the MasterBlaster™ or the ByteBlasterMV™ communications cables) can easily configure a working device in minutes.

After a successful compilation, the Quartus Compiler generates one or more programming files, which the Programmer can use to program or configure one or more devices. You can download configuration data into APEX 20K devices through the MasterBlaster or the ByteBlasterMV communications cable.

You can configure APEX 20K devices in either Passive Serial configuration mode or in JTAG mode. You can program one or more configuration devices in JTAG mode. These devices can be programmed in the top-to-bottom order specified in the programming list of the Programmer window. However, this tutorial explains how to program only a single device.

This session assumes that the project has already been compiled.

Session 18: Program an Altera Device

The following steps are covered in this session:

1. Open the Programmer window.
2. Set up a Passive Serial chain.
3. Configure the device.
4. Change programming modes.
5. Add a device to a chain.

1. Open the Programmer Window

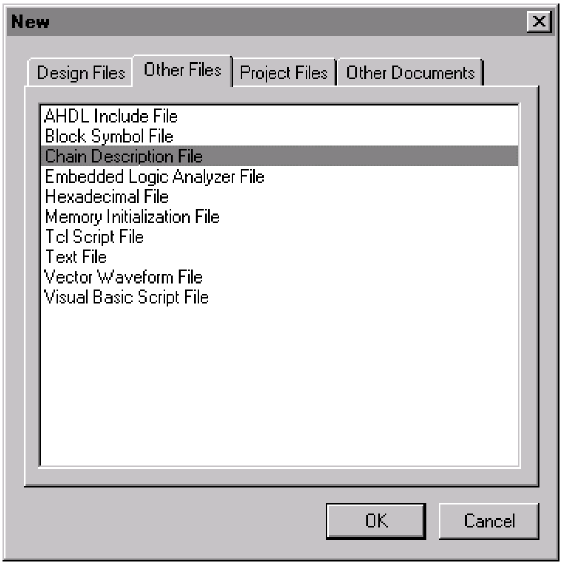
When you are getting ready to program or configure a device, you need to open the Programmer and create a Chain Description File (**.cdf**) that stores device name, device order, and programming and hardware setup information. You can use CDFs to program or configure one or more devices in a JTAG chain or a Passive Serial chain.

To open the Programmer window and create a CDF, follow these steps:

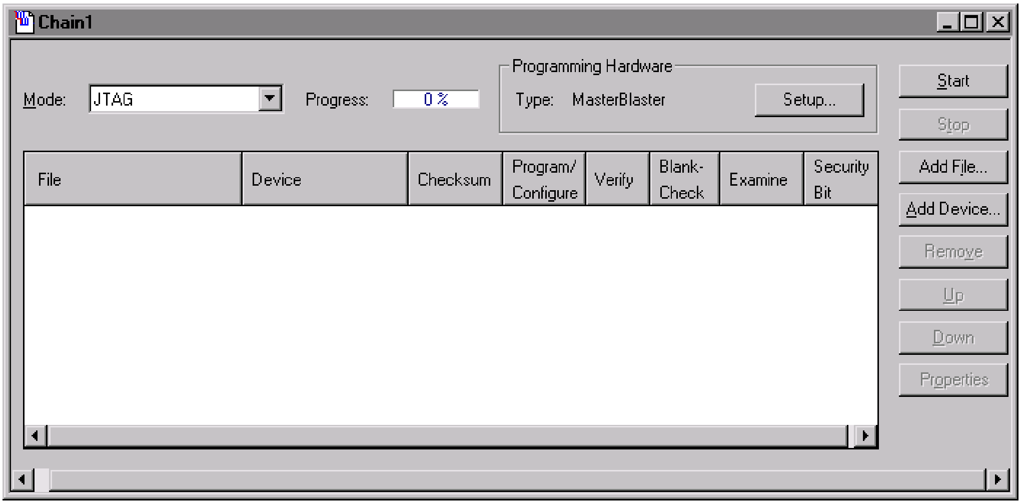
1. Choose **New** (File menu).

2. In the **New** dialog box, click the **Other Files** tab.

3. From the **Other Files** list, select **Chain Description File**, as shown in the following illustration:



4. To create the Chain Description File, click **OK**. The Programmer window opens a blank CDF, as shown in the following illustration:



Mode: JTAG

Progress: 0 %

Type: MasterBlaster

5. Choose **Save As** (File menu).

6. In the **Save As** dialog box, type `fir_filter.cdf` in the **File name** box.

7. Make sure that in the **Save as type** list, **Chain Description File** is selected.

8. Click **Save**.

You can also open the Programmer window by choosing **Open Programmer** (Processing menu).

2. Set Up a Passive Serial Chain

To set up configuration for a single device in a Passive Serial “chain,” follow these steps:

1. In the **Mode** list of the Programmer window, select **Passive Serial**.

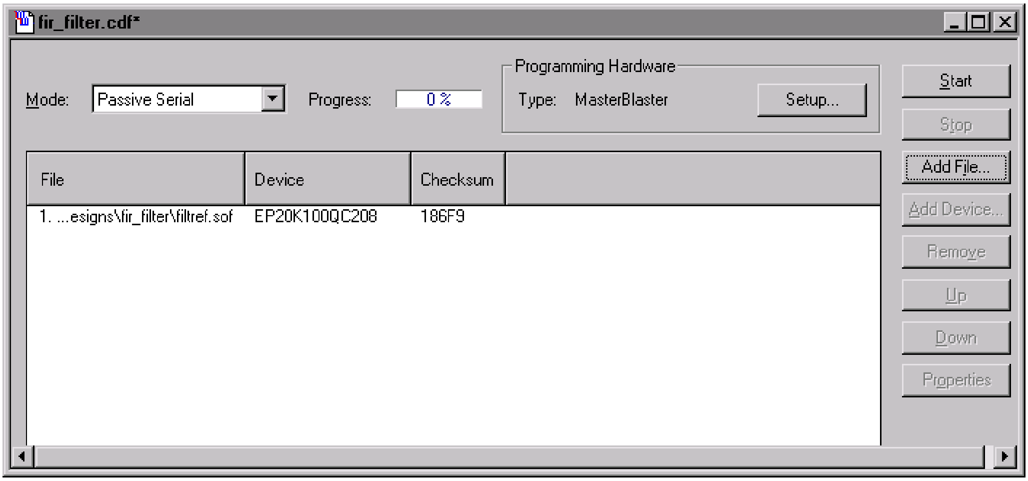
2. Under **Programming Hardware**, click **Setup**. The **Hardware Setup** dialog box opens.

3. In the **Hardware Type** list, select either **ByteBlasterMV** or **MasterBlaster** and, if necessary, specify the port and baud rate. Click **OK**.

4. Click **Add File**.

5. In the **Select File** dialog box, specify the **filtref.sof** file in the **File name** box.

6. Click **Open**. See the following illustration:



fir_filter.cdf*

Mode: Passive Serial

Progress: 0 %

Programming Hardware

Type: MasterBlaster

File	Device	Checksum
1. ...esigns\fir_filter\filtref.sof	EP20K100QC208	186F9

7. Choose **Save** (File menu).

If you want to program multiple devices, you can repeat steps 1 through 5 to add additional devices. However, for the purposes of this tutorial, only one programming file is necessary.

3. Configure the Device

To configure the device(s), follow these steps:

1. To attach the appropriate communications cable to your PC or UNIX workstation, perform one of the following steps:

✓ Attach the ByteBlasterMV or MasterBlaster cable to your PC. Refer to the *Quartus Installation & Licensing for PCs* manual for more information.

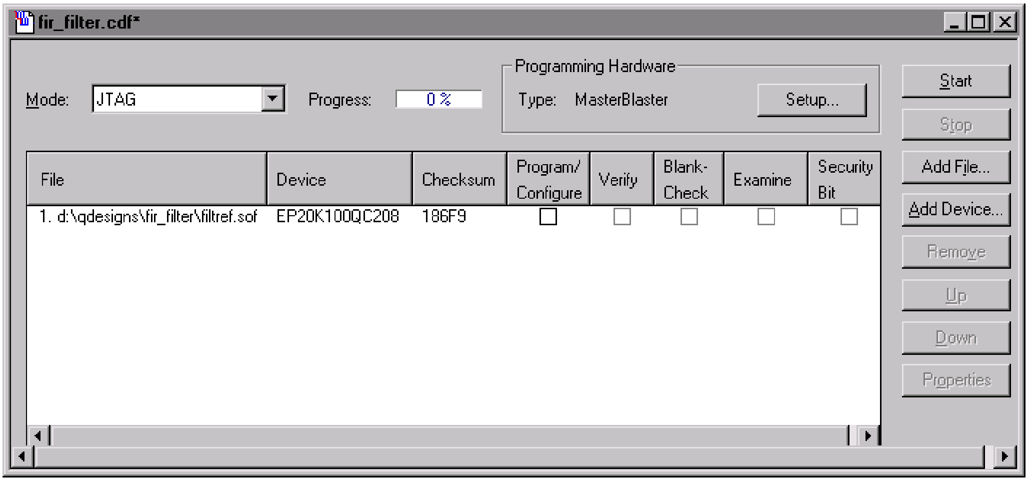
or

✓ Attach the MasterBlaster cable to your UNIX workstation. Refer to the *Quartus Installation & Licensing for UNIX Workstations* manual for more information.

2. Click **Start** in the Programmer window. A message appears when the configuration is complete.

4. Change Programming Modes

When configuring APEX20K devices, you can switch between JTAG and Passive Serial programming modes. If you select JTAG mode, both configuration and programming options appear in the window. However, only the **Program/Configure** option is available for APEX20K and devices. All other options that appear in JTAG mode, are available only for configuration devices, such as the EPC2 configuration device. See the following illustration:



fir_filter.cdf*

Mode: JTAG

Progress: 0 %

Programming Hardware

Type: MasterBlaster

File	Device	Checksum	Program/ Configure	Verify	Blank-Check	Examine	Security Bit
1. d:\qdesigns\fir_filter\filtref.sof	EP20K100QC208	186F9	□	□	□	□	□

5. Add a Device to a Chain

You can add devices to a JTAG or Passive Serial chain using the Programmer window. When you add a configuration device to the chain, you can turn on the **Examine** option to load the programming data from the device into a temporary buffer. Then, you can save the programming data to a Programmer Object File (**.pof**).

If you want the Programmer to bypass a device in the chain, that is, to allow data to pass through it without performing any programming action, turn off all programming options—**Program**, **Verify**, **Blank-Check**, and **Examine**—for the device.

To add a device to a JTAG chain, follow these steps:

1. In the Programmer window, click **Add Device**.
2. In the **Select Device** dialog box, select the device you want to add in the **Devices** list.
3. Click **OK**.
4. Choose **Save** (File menu).

☞ If you want to add additional devices, you can repeat steps 1 through 3. However, for the purposes of this tutorial, only one device is necessary.

For more information about using the Programmer, refer to Quartus Help.

Contacting Altera

If you have any additional questions about Altera products, contact Altera for technical support and product information.

Technical Support

If you need technical support, call or fax the Altera Applications Department Monday through Friday.

Tel:	(800) 800-EPLD	(6:00 a.m. to 6:00 p.m. Pacific Time)
	(408) 544-7000	(7:30 a.m. to 5:30 p.m. Pacific Time)
Fax:	(408) 544-6401	

You can also e-mail the Altera Applications Department at **support@altera.com.**

In addition, you can visit the **Atlas**$^{\text{SM}}$ online solutions database, which is located at **http://www.altera.com**.

For additional technical support for the Quartus software, choose **Altera on the Web** > **Quartus Home Page** (Help menu), or point your browser to the Quartus support page at **https://websupport.altera.com**. This web site provides information on how to register your software and obtain license information, and provides other support information.

Product Information

If you need the latest Altera product information or literature, use the Altera web site, which is available 24 hours a day, seven days a week.

Web site: **http://www.altera.com**

Go to “Contacting Altera” in Quartus Help for complete information on Altera technical support services.

Revision History

The information contained in the Quartus Tutorial version 1999.10 revision 2 supersedes information published in previous versions.

On pages 73, 74, 79, and 85 minor text changes were made.

On page 55, step 2 was updated to reflect a user interface change. The related graphic on page 56 was also updated.