



# **Complex Multiplier (ALTMULT\_COMPLEX)**

---

## **Megafunction User Guide**



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

Software Version: 8.0  
Document Version: 1.0  
Document Date: May 2008

Copyright © 2008 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001

UG-01033-1.0



---

<b>About This User Guide</b> .....	<b>v</b>
Revision History .....	v
Referenced Documents .....	v
How to Contact Altera .....	vi
Typographic Conventions .....	vi
<b>Chapter 1. About this Megafunction</b>	
Device Family Support .....	1-1
Introduction .....	1-1
Features .....	1-1
General Description .....	1-2
Complex Multiplication .....	1-2
Conventional Representation .....	1-3
Canonical Representation .....	1-3
Resource Usage and Performance .....	1-4
<b>Chapter 2. Getting Started</b>	
Software and System Requirements .....	2-1
MegaWizard Plug-In Manager Customization .....	2-1
MegaWizard Plug-In Manager Page Descriptions .....	2-2
Instantiating Megafunctions in HDL Code or Schematic Designs .....	2-9
Generating a Netlist for EDA Tool Use .....	2-10
Using the Port and Parameter Definitions .....	2-10
Identifying a Megafunction after Compilation .....	2-11
Simulation .....	2-11
Quartus II Software Simulation .....	2-11
EDA Tool Simulation .....	2-12
Design Example 1: Multiplication of 8-Bit Complex Numbers Using Canonical Representation ...	
2-12	
Design Files .....	2-12
Configuration Settings .....	2-13
Functional Simulation in the Quartus II Software .....	2-13
Comparison of Resource Usage .....	2-15
Functional Simulation in the ModelSim-Altera Simulator .....	2-19
Conclusion .....	2-20
<b>Chapter 3. Specifications</b>	
Ports and Parameters .....	3-1





# About this User Guide

**Revision History** The following table shows the revision history for this user guide.

Date and Document Version	Changes Made	Summary of Changes
May 2008 v1.0	Initial release.	—

## Referenced Documents

This user guide references the following documents:

- *Configuring Cyclone II Devices* chapter in the Cyclone II Device Handbook
- *Configuring Stratix II and Stratix II GX Devices* chapter in volume 2 of the Stratix II Device Handbook
- *Configuring Stratix III Devices* chapter in volume 1 of the Stratix III Device Handbook
- *Cyclone II Architecture* chapter in the Cyclone II Device Handbook
- *DSP Blocks in Stratix and Stratix GX Devices* chapter in volume 2 of the Stratix Device Handbook
- *DSP Blocks in Stratix II and Stratix II GX Devices* chapter in volume 2 of the Stratix II Device Handbook
- *DSP Blocks in Stratix III Devices* chapter in volume 1 of the Stratix III Device Handbook
- *Embedded Multipliers in Cyclone II Devices* chapter in the Cyclone II Device Handbook
- *Implementing High-Performance DSP Functions in Stratix and Stratix GX Devices* chapter in volume 2 of the Stratix Device Handbook
- *Introduction* chapter in the Cyclone II Device Handbook
- *Introduction* chapter in volume 1 of the Stratix Device Handbook
- *Introduction* chapter in volume 1 of the Stratix II Device Handbook
- *Multiplier-Adder Megafunction User Guide (ALTMULT\_ADD)*
- *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*
- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*
- *Simulation* section in volume 3 of the *Quartus II Handbook*
- *Stratix Architecture* chapter in volume 1 of the Stratix Device Handbook

- *Stratix II Architecture* chapter in volume 1 of the Stratix II Device Handbook
- *Stratix III Device Family Overview* chapter in volume 1 of the Stratix III Device Handbook
- *Synthesis* section in volume 1 of the *Quartus II Handbook*

## How to Contact Altera

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

Contact (1)	Contact Method	Address
Technical support	Website	<a href="http://www.altera.com/support">www.altera.com/support</a>
Technical training	Website	<a href="http://www.altera.com/training">www.altera.com/training</a>
	Email	<a href="mailto:custrain@altera.com">custrain@altera.com</a>
Product literature	Website	<a href="http://www.altera.com/literature">www.altera.com/literature</a>
Non-technical support (General) (Software Licensing)	Email	<a href="mailto:nacomp@altera.com">nacomp@altera.com</a>
	Email	<a href="mailto:authorization@altera.com">authorization@altera.com</a>








*Note to table:*

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

## Typographic Conventions

This document uses the typographic conventions shown in the following table.

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

Visual Cue	Meaning
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: <code>data1</code> , <code>tdi</code> , <code>input</code> . Active-low signals are denoted by suffix <code>n</code> , e.g., <code>resetn</code> .  Anything that must be typed exactly as it appears is shown in Courier type. For example: <code>c:\qdesigns\tutorial\chiptrip.gdf</code> . Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword <code>SUBDESIGN</code> ), as well as logic function names (e.g., <code>TRI</code> ) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets are used in a list of items when the sequence of the items is not important.
	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information about a particular topic.



## Device Family Support

The Complex Multiplier (ALTMULT\_COMPLEX) megafunction supports the following target Altera® device families:

- Arria™ GX
- Cyclone® III
- Cyclone II
- HardCopy® II
- HardCopy Stratix®
- Stratix IV
- Stratix III
- Stratix II
- Stratix II GX
- Stratix
- Stratix GX

## Introduction

As design complexities increase, the use of vendor-specific intellectual property (IP) blocks has become a common design methodology. Altera provides parameterizable megafunctions that are optimized for Altera device architectures. Using megafunctions instead of coding your own logic saves valuable design time. Additionally, the Altera-provided functions may offer more efficient logic synthesis and device implementation. You can scale the size of the megafunction by setting parameters.

## Features

The ALTMULT\_COMPLEX megafunction implements the multiplication of two complex numbers and offers many additional features, including the following:


- Parameterizable input data widths ranging from 1 to 256 bits
- Output result widths of up to 256 bits
- Two representations of implementation: canonical representation and conventional representation
- Support for both signed and unsigned data representation formats
- Active-high asynchronous clear and clock-enable control inputs
- Option to add the extra latency required to pipeline the output of the multiplier

## General Description

The ALTMULT\_COMPLEX megafunction simplifies the process of multiplying complex numbers. The megafunction offers the following two implementation modes:

- Canonical  
You can use the canonical representation for the following:
  - Input data widths of less than 18 bits
  - All Altera devices listed in “[Device Family Support](#)” except Stratix III devices.
- Conventional  
You can use the conventional representation for the following:
  - Input data widths of 18 bits or more
  - Stratix III devices

With the conventional representation, you can use the ALTMULT\_ADD megafunction to implement the complex multiplier by instantiating two multipliers.

 For more information about the ALTMULT\_ADD megafunction, refer to the [Multiplier-Adder Megafunction User Guide \(ALTMULT\\_ADD\)](#).

## Complex Multiplication

Complex numbers are numbers in the form of the following equation:

$$a + ib$$

where:

- $a$  and  $b$  are real numbers
- $i$  is an imaginary unit that equals the square root of  $-1$ :  $\sqrt{-1}$

Two complex numbers  $x = a + ib$  and  $y = c + id$  are multiplied, as shown in [Equation 1](#) and [Equation 2](#):

$$(1) \quad xy = (a + ib)(c + id)$$

$$= ac + ibc + iad - bd$$

$$(2) \quad = (ac - bd) + i(ad + bc)$$

## Conventional Representation

From Equation 1, the multiplication of two complex numbers can be represented in two parts: real and imaginary. The  $xy\_real$  variable in Equation 3 represents the real part, and the  $xy\_imaginary$  variable in Equation 4 represents the imaginary part. Both equations are derived from Equation 2.

$$(3) \quad xy\_real = ac - bd$$

$$(4) \quad xy\_imaginary = ad + bc$$

Table 1–1 shows the number of resources required to represent the multiplication of two complex numbers in the conventional representation. Two ALTMULT\_ADD megafunction blocks are instantiated to implement this representation—one block to generate the output of the real value and another block to generate the output of the imaginary value.

Representation	Multiplier	Adder	Subtractor
Conventional	4	1	1

For Stratix III devices, the conventional representation improves resource efficiency as the ALTMULT\_ADD megafunction is inferred in the ALTMULT\_COMPLEX megafunction for the device.

## Canonical Representation

From Equation 1, the multiplication of two complex numbers can be represented in two parts: real and imaginary. The  $xy\_real$  variable in Equation 5 represents the real part, and the  $xy\_imaginary$  variable in Equation 6 represents the imaginary part. Both equations are derived from Equation 2.

$$(5) \quad xy\_real = ac - bd$$

$$= ac - bd + (ad - bc) - (ad - bc)$$

$$= (ac - ad + bc - bd) + (ad - bc)$$

$$= ((a + b)(c - d)) + (ad - bc)$$

$$(6) \quad xy\_imaginary = ad + bc$$



The canonical representation is available for all Altera devices listed in “[Device Family Support](#)” except Stratix III devices.

Table 1–2 shows the number of resources required to represent the multiplication of two complex numbers in the canonical representation. The multiplication requires the instantiation of one comparator and three multiplexers.

Representation	Multiplier	Adder	Subtractor
Canonical	3	3	4



For more information about the multiply or add mode in the DSP blocks of Stratix devices and details on using accumulators in the finite impulse response (FIR) filter applications, refer to the following literature:

- *Stratix II Architecture* chapter in volume 1 of the Stratix II Device Handbook
- *DSP Blocks in Stratix and Stratix GX Devices* chapter in volume 2 of the Stratix Device Handbook
- *Implementing High-Performance DSP Functions in Stratix and Stratix GX Devices* chapter in volume 2 of the Stratix Device Handbook

## Resource Usage and Performance

For more information about the architecture of DSP blocks, embedded multipliers, and hardware conversion process, refer to the following literature:

- *Stratix III Device Family Overview* chapter in volume 1 of the Stratix III Device Handbook
- *DSP Blocks in Stratix III Devices* chapter in volume 1 of the Stratix III Device Handbook
- *Configuring Stratix III Devices* chapter in volume 1 of the Stratix III Device Handbook
- *Introduction* chapter in volume 1 of the Stratix II Device Handbook
- *Stratix II Architecture* chapter in volume 1 of the Stratix II Device Handbook
- *DSP Blocks in Stratix II and Stratix II GX Devices* chapter in volume 2 of the Stratix II Device Handbook
- *Configuring Stratix II and Stratix II GX Devices* chapter in volume 2 of the Stratix II Device Handbook
- *Introduction* chapter in volume 1 of the Stratix Device Handbook
- *Stratix Architecture* chapter in volume 1 of the Stratix Device Handbook
- *DSP Blocks in Stratix and Stratix GX Devices* chapter in volume 2 of the Stratix Device Handbook

- *Implementing High-Performance DSP Functions in Stratix and Stratix GX Devices* chapter in volume 2 of the Stratix Device Handbook
- *Introduction* chapter in the Cyclone II Device Handbook
- *Cyclone II Architecture* chapter in the Cyclone II Device Handbook
- *Embedded Multipliers in Cyclone II Devices* chapter in the Cyclone II Device Handbook
- *Configuring Cyclone II Devices* chapter in the Cyclone II Device Handbook
- Other related device documentation available on the Literature page of the Altera website ([www.altera.com](http://www.altera.com)).

Table 1–3 summarizes the resource usage of the ALTMULT\_COMPLEX megafunction for Stratix III and Cyclone III devices.

<b>Table 1–3. ALTMULT_COMPLEX Megafunction Resource Usage for Stratix III and Cyclone III Devices</b>				
<i>Note (1)</i>				
<b>Device Family</b>	<b>Representation (2)</b>	<b>Input Data Width (bits)</b>	<b>Logic Usage (Dedicated Logic Registers)</b>	<b>DSP Blocks</b>
Stratix III	Conventional	18	36	4
Cyclone III	Canonical	9	156	0
	Conventional	9	90	4
	Conventional	18	180	4

**Notes to Table 1–3:**

- (1) You can get the resource information from the MegaWizard® Plug-In Manager. The information in this table is valid and accurate in the Quartus II software version 8.0.
- (2) Specify the representation on page 3 of the MegaWizard Plug-In Manager for the ALTMULT\_COMPLEX megafunction. For Stratix III devices and input data widths of 18 bits and above, use only the conventional representation.



### Software and System Requirements

The instructions in this section require the following software:

- Quartus® II software version 8.0 or later
- For operating system support information, refer to:  
[www.altera.com/support/software/os\\_support/oss-index.html](http://www.altera.com/support/software/os_support/oss-index.html)

### MegaWizard Plug-In Manager Customization

The MegaWizard® Plug-In Manager creates or modifies design files that contain custom megafunction variations, which can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard that allows you to specify options for the Complex Multiplier (ALTMULT\_COMPLEX) megafunction features in your design.

Start the MegaWizard Plug-In Manager in one of the following ways:

- On the Tools menu, click **MegaWizard Plug-In Manager**.
- When working in the Block Editor, on the Edit menu, click **Insert Symbol as Block**, or right-click in the Block Editor, point to **Insert**, and click **Symbol as Block**. In the **Symbol** window, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:  
`qmegawiz ←`

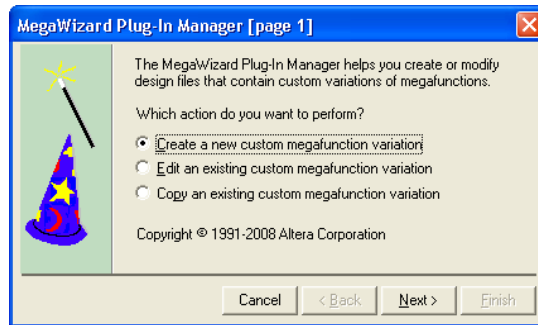
## MegaWizard Plug-In Manager Page Descriptions

This section provides descriptions of the options available on the individual pages of the ALTMULT\_COMPLEX wizard.

On page 1 of the MegaWizard Plug-In Manager, you can select **Create a new custom megafunction variation**, **Edit an existing custom megafunction variation**, or **Copy an existing custom megafunction variation** (Figure 2-1).

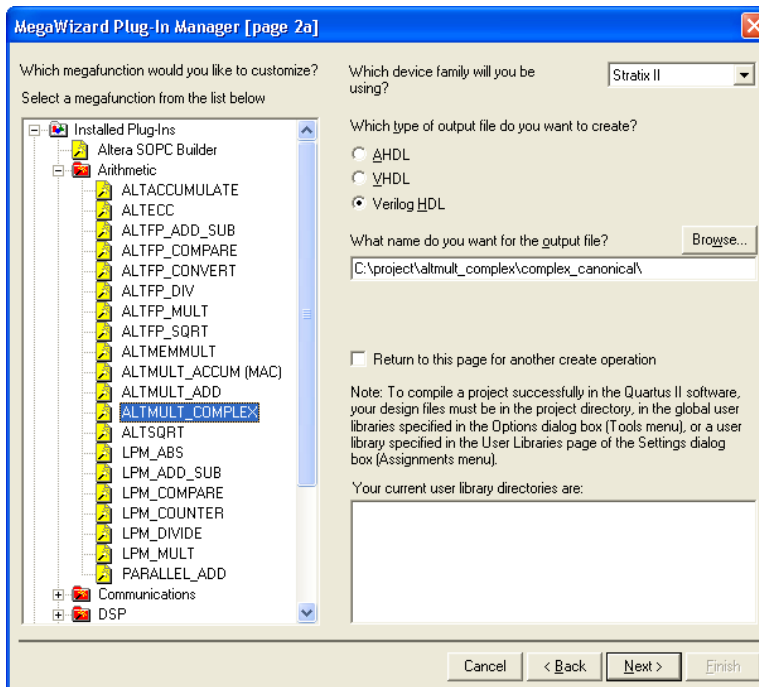
---

**Figure 2-1. MegaWizard Plug-In Manager [page 1]**



On page 2a of the MegaWizard Plug-In Manager, specify the megafunction, the device family to use, the type of output file to create, and the name of the output file (Figure 2-2). Choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v) as the output file type.

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



On page 3 of the MegaWizard Plug-In Manager, select the width of the input and output buses. Specify whether the input buses are signed or unsigned (Figure 2-3).

Figure 2-3. ALTMULT\_COMPLEX MegaWizard Plug-In Manager [page 3 of 6]

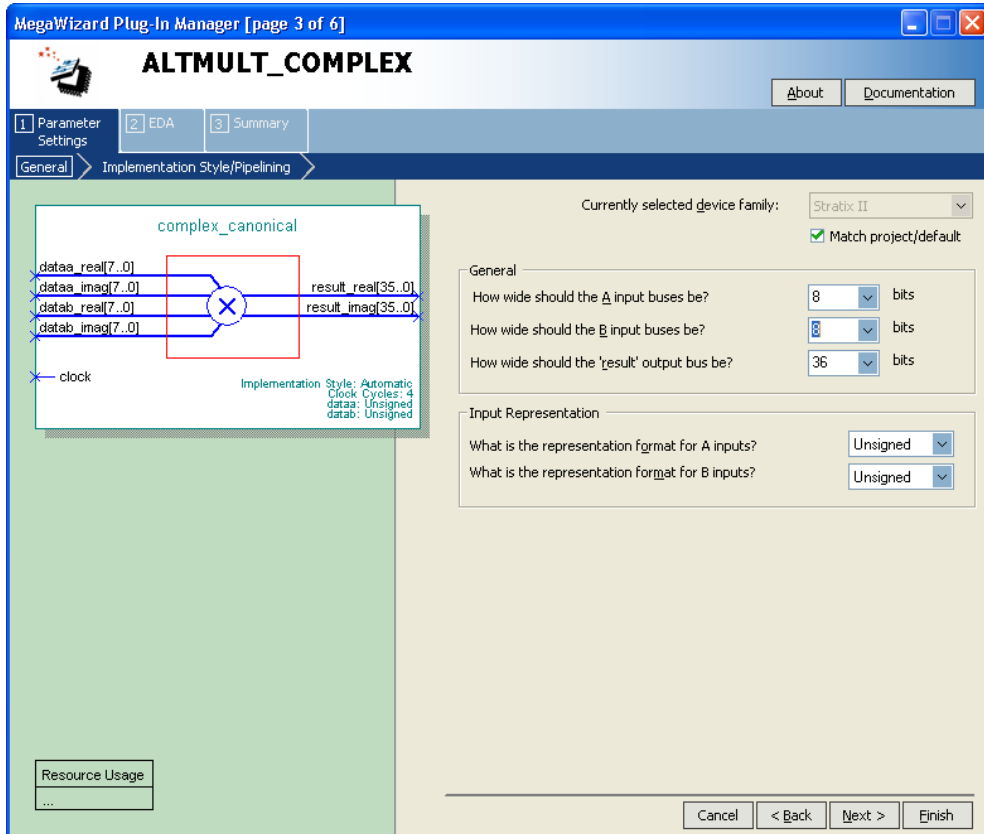


Table 2-1 shows the options available on page 3 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager.

<b>Configuration Setting</b>	<b>Description</b>
Currently selected device family	Select the device family you want to use. Turn off <b>Match project/default</b> to change the device family if it is different from the default.
How wide should the A input buses be? How wide should the B input buses be?	Select the width of the input buses. You can choose from 1 to 256 bits.
How wide should the result output bus be?	Select the width of the output bus. You can choose from 1 to 256 bits.
What is the representation format for A inputs? What is the representation format for B inputs?	Select <b>Signed</b> or <b>Unsigned</b> for inputs A and B.

On page 4 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager, specify the implementation style and select the optional inputs (Figure 2-4).

Figure 2-4. ALTMULT\_COMPLEX MegaWizard Plug-In Manager [page 4 of 6]

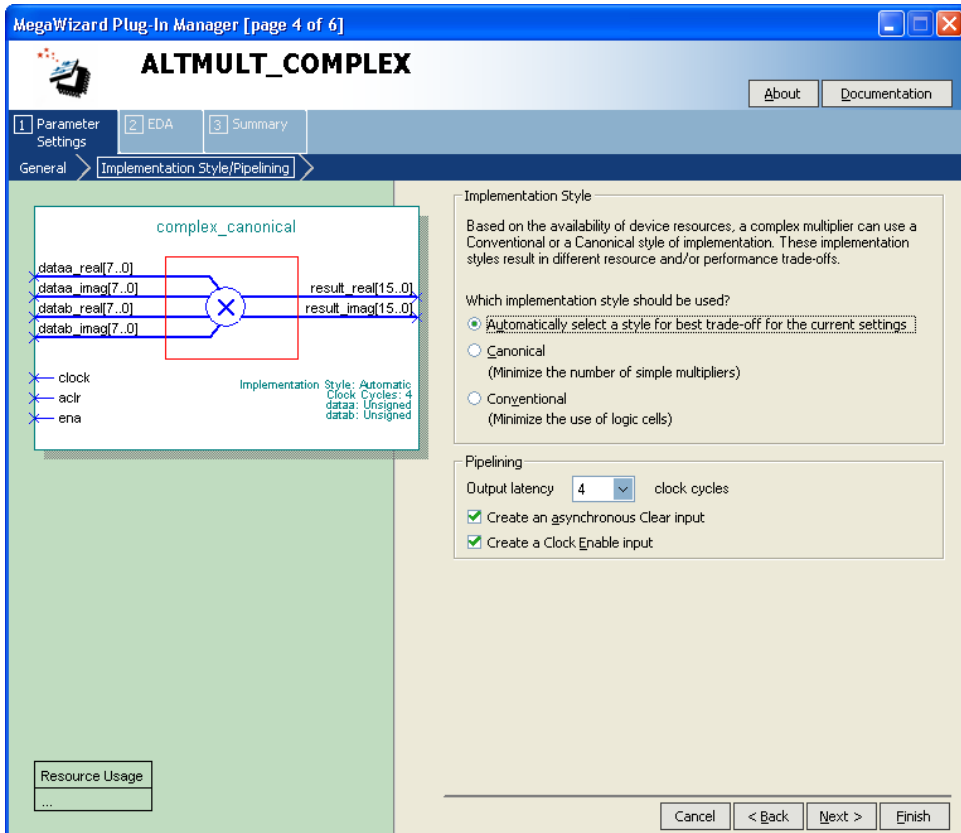
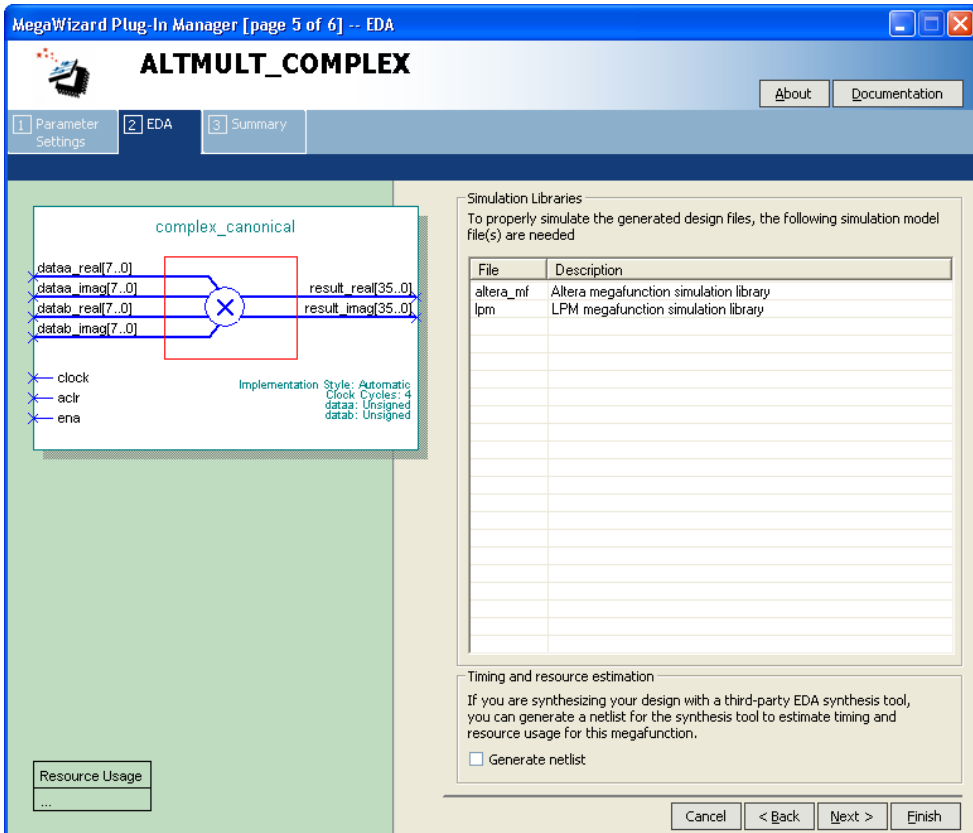


Table 2–2 shows the options available on page 4 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager.

<b>Configuration Setting</b>	<b>Description</b>
Which implementation style should be used?	<p>Select an implementation style for the complex multiplier.</p> <p>The <b>Automatically select a style for best trade-off for the current settings</b> option chooses the best implementation style based on the device family and input data widths specified on page 3 of the MegaWizard Plug-In Manager.</p> <p>The <b>Canonical (Minimize the number of simple multipliers)</b> option is used for non-Stratix III devices or when the input data width is less than 18 bits.</p> <p>The <b>Conventional (Minimize the use of logic cells)</b> option is used for Stratix III devices or when the input data width is 18 bits and above.</p>
Pipelining	<p>Select the output latency. You can choose a value between 0 and 14 clock cycles.</p> <p>Select <b>Create an asynchronous Clear input</b> to create the asynchronous clear (<code>aclr</code>) port. Use the <code>aclr</code> port to reset the megafunction asynchronously.</p> <p>Select <b>Create a Clock Enable input</b> to create the clock-enable (<code>ena</code>) port. Use the optional <code>ena</code> port to enable the clock.</p>

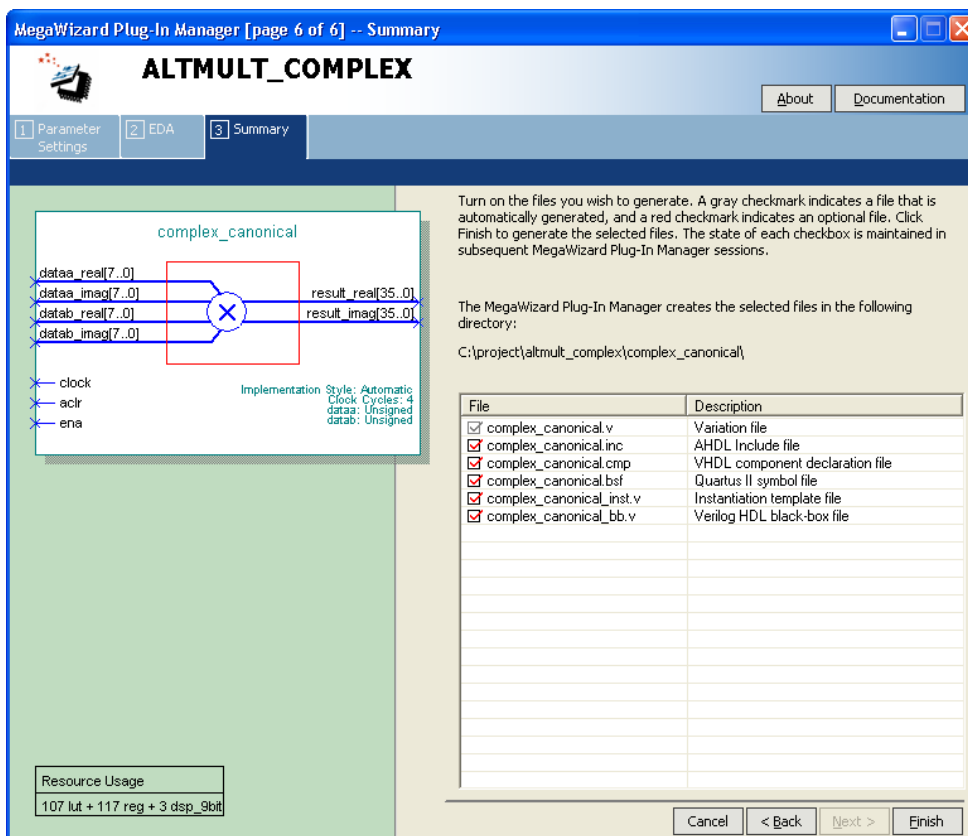
On page 5 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager, you can choose to generate a resource usage and timing estimation netlist (Figure 2–5).

Figure 2–5. ALTMULT\_COMPLEX MegaWizard Plug-In Manager [page 5 of 6]



On page 6 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager, specify the types of files to be generated. Choose from the AHDL include file (*<function name>.inc*), VHDL component declaration file (*<function name>.cmp*), Quartus II symbol file (*<function name>.bsf*), instantiation template file (*<function name>.inst.v*), Verilog HDL black-box declaration file (*<function name>.bb.v*), and Quartus II IP Advisor file (*<function name>.qip*). If you selected **Generate netlist** on page 5, the file for that netlist is also available. A gray checkmark indicates a file that is automatically generated and a red checkmark indicates an optional file (Figure 2–6).

Figure 2–6. ALTMULT\_COMPLEX MegaWizard Plug-In Manager [page 6 of 6]



For more information about the ports and parameters for the ALTMULT\_COMPLEX megafunction, refer to [Chapter 3, Specifications](#).

## Instantiating Megafunctions in HDL Code or Schematic Designs

When you use the MegaWizard Plug-In Manager to customize and parameterize a megafunction, it creates a set of output files that allows you to instantiate the customized function in your design. Depending on the language you choose in the MegaWizard Plug-In Manager, the wizard instantiates the megafunction with the correct parameter values and generates a megafunction variation file (wrapper file) in Verilog HDL (.v), VHDL (.vhd), or AHDL (.tdf), along with other supporting files.

The MegaWizard Plug-In Manager provides options to create the following files:

- A sample instantiation template for the language of the variation file (`_inst.v`, `_inst.vhd`, or `_inst.tdf`)
- Component Declaration File (`.cmp`) that can be used in VHDL Design Files
- ADHL Include File (`.inc`) that can be used in Text Design Files (`.tdf`)
- Quartus II Block Symbol File (`.bsf`) that can be used in schematic designs
- Verilog HDL module declaration file that can be used when instantiating the megafunction as a black box in a third-party synthesis tool (`_bb.v`)



For more information about the wizard-generated files, refer to Quartus II Help or to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

### Generating a Netlist for EDA Tool Use

If you use a third-party EDA synthesis tool, you can instantiate the megafunction variation file as a black box for synthesis. Use the VHDL component declaration or Verilog HDL module declaration black-box file to define the function in your synthesis tool, and then include the megafunction variation file in your Quartus II project.

If you enable the option to generate a synthesis area and timing estimation netlist in the MegaWizard Plug-In Manager, the wizard generates an additional netlist file (`_syn.v`). The netlist file is a representation of the customized logic used in the Quartus II software. The file provides the connectivity of the architectural elements in the megafunction but may not represent true functionality. This information enables certain third-party synthesis tools to better report area and timing estimates. In addition, synthesis tools can use the timing information to focus timing-driven optimizations and improve the quality of results.



For more information about using megafunctions in your third-party synthesis tool, refer to the appropriate chapter in the *Synthesis* section in volume 1 of the *Quartus II Handbook*.

### Using the Port and Parameter Definitions

Instead of using the MegaWizard Plug-In Manager, you can instantiate the megafunction directly in your Verilog HDL, VHDL, or AHDL code by calling the megafunction and setting its parameters as you would any other module, component, or subdesign.



Altera strongly recommends that you use the MegaWizard Plug-In Manager for complex megafunctions. The MegaWizard Plug-In Manager ensures that you set all megafunction parameters properly.

For a list of the megafunction ports and parameters, refer to [Chapter 3, Specifications](#).

## Identifying a Megafunction after Compilation

During compilation with the Quartus II software, analysis and elaboration are performed to build the structure of your design. To locate your megafunction in the **Project Navigator** window, expand the compilation hierarchy and find the megafunction by its name.

To search for node names within the megafunction (using the Node Finder), click **Browse** in the **Look in** box and select the megafunction in the **Hierarchy** box.

## Simulation

The Quartus II Simulator provides an easy-to-use, integrated solution for performing simulations. The following sections describe the simulation options.

### Quartus II Software Simulation

With the Quartus II Simulator, you can perform two types of simulations: functional and timing. A functional simulation enables you to verify the logical operation of your design without taking into consideration the timing delays in the FPGA. This simulation is performed using only your RTL code. When performing a functional simulation, add only signals that exist before synthesis. You can find these signals with the Registers: Pre-Synthesis, Design Entry, or Pin filters in the Node Finder. The top-level ports of megafunctions are found using these three filters.

In contrast, the timing simulation in the Quartus II software verifies the operation of your design with annotated timing information. This simulation is performed using the post place-and-route netlist. When performing a timing simulation, add only signals that exist after place-and-route. These signals are found with the post-compilation filter of the Node Finder. During synthesis and place-and-route, the names of RTL signals change. Therefore, it may be difficult to find signals from your megafunction instantiation in the post-compilation filter.

To preserve the names of your signals during the synthesis and place-and-route stages, use the synthesis attributes `keep` or `preserve`. These are Verilog HDL and VHDL synthesis attributes that direct analysis

and synthesis to keep a particular wire, register, or node intact. Use these synthesis attributes to keep a combinational logic node so you can observe the node during simulation.



For more information about these attributes, refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

### EDA Tool Simulation

The *Quartus II Handbook* chapters describe how to perform functional and gate-level timing simulations that include the megafunctions, with details about the files that are needed and the directories where the files are located.



Depending on which simulation tool you are using, refer to the appropriate chapter in the *Simulation* section in volume 3 of the *Quartus II Handbook*.

## Design Example 1: Multiplication of 8-Bit Complex Numbers Using Canonical Representation

This design example uses the MegaWizard Plug-In Manager to configure the ALTMULT\_COMPLEX megafunction. The design uses a Stratix II device to implement a complex multiplier with an 8-bit input data width using the canonical representation. The ModelSim®-Altera software is used for the simulation.

The example also compares the resource usage between the canonical and conventional implementations for a design with an 8-bit input data width. The canonical representation requires less resources for designs with less than 18 bits of input data width, or designs involving non-Stratix III devices.

### Design Files

The example design files are available in the User Guides section on the Literature page of the Altera website ([www.altera.com](http://www.altera.com)).

The following functional simulations are available for the design example:

- “Functional Simulation in the Quartus II Software” on page 2–13
- “Functional Simulation in the ModelSim-Altera Simulator” on page 2–19

Download the respective design example for the Quartus II or ModelSim-Altera software before running the simulation.

## Configuration Settings

This design example configures the ALTMULT\_COMPLEX megafunction for the canonical representation with an input data width of 8 bits.

Table 2–3 shows the ALTMULT\_COMPLEX MegaWizard Plug-In Manager configuration settings that are used to create this design example. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 2–3. Click **Next** to advance from one page to the next.

<b>Configuration Settings</b>	<b>Value</b>
Device family	Stratix II
Output file type	Verilog
Bus width for data A and data B inputs	8 bits
Bus width for result output	16 bits
Representation format for data A and data B inputs	Unsigned
Output latency	4
Implementation style	Canonical
Create an asynchronous clear port	Yes
Create a clock enable port	Yes
Generate netlist	No

## Functional Simulation in the Quartus II Software

Run the Quartus II functional simulation to display the functional behavior waveform of the design example. The design files are already configured and compiled.

Set up and run the Quartus II Simulator by performing the following steps:

1. Open the **altnult\_complex\_DesignExample\_ex1.zip** project and extract the **complex\_canonical.qar** file.
2. In the Quartus II software, open the **complex\_canonical.qar** file and restore the archived file into your working directory.
3. On the Processing menu, click **Start Compilation**.

4. When the **Full compilation was successful** message box appears, click **OK**.
5. At the Compilation Report window, expand the Fitter menu hierarchy, followed by the Resource Section menu hierarchy. Select **DSP Block Usage Summary**. The number of DSP blocks used is displayed (Figure 2-7).

**Figure 2-7. Resource Usage for Canonical Representation**

Fitter DSP Block Usage Summary				
	Statistic	Number Used	Available per Block	Maximum Available
1	Simple Multipliers (9-bit)	3	8	96
2	Simple Multipliers (18-bit)	0	4	48
3	Simple Multipliers (36-bit)	0	1	12
4	Multiply Accumulators (18-bit)	0	2	24
5	Two-Multipliers Adders (9-bit)	0	4	48
6	Two-Multipliers Adders (18-bit)	0	2	24
7	Four-Multipliers Adders (9-bit)	0	2	24
8	Four-Multipliers Adders (18-bit)	0	1	12
9	Dynamic DSP Blocks	0	1	12
10	DSP Blocks	1	--	12
11	DSP Block 9-bit Elements	3	8	96
12	Signed Multipliers	0	--	--
13	Unsigned Multipliers	3	--	--
14	Mixed Sign Multipliers	0	--	--
15	Variable Sign Multipliers	0	--	--
16	Dedicated Shift Register Chains	0	--	--

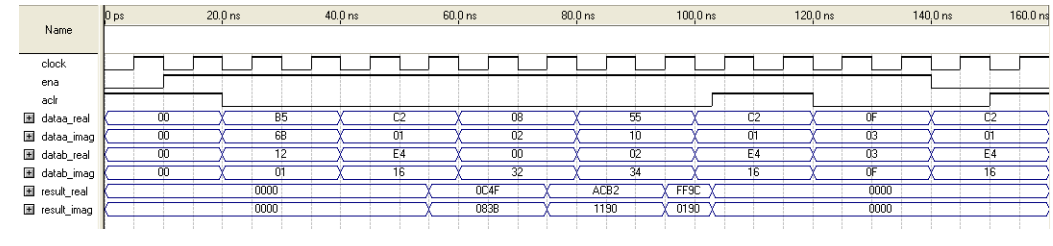


The resource usage values are correct at the time of publication. For updated results, refer to the Compilation Report of the latest version of the Quartus II software.

6. On the Processing menu, click **Generate Functional Simulation Netlist**.
7. When the **Functional Simulation Netlist Generation was successful** message box appears, click **OK**.
8. On the Processing menu, click **Start Simulation**, or, on the toolbar, click the **Start Simulation** button.
9. When the **Simulator was successful** message box appears, click **OK**.

- In the Simulation Waveforms window, view the simulation output waveforms to verify the results (Figure 2–8).

**Figure 2–8. Functional Simulation in the Quartus II Software**

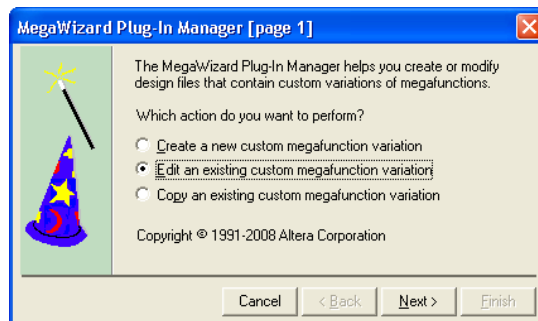


## Comparison of Resource Usage

To compare the resource usage between the canonical and conventional representations, edit the design file in the MegaWizard Plug-In Manager by performing the following steps:

- The project file of Design Example 1 must be open before you perform the following steps. Otherwise, browse to your working directory and open the **complex\_canonical.ppf** file.
- On the Tools menu, click **MegaWizard Plug-In Manager**. The MegaWizard Plug-In Manager dialog box appears (Figure 2–9).

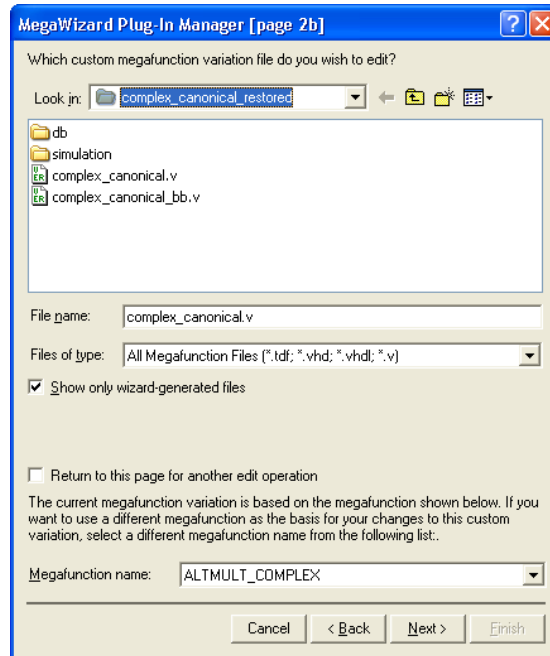
**Figure 2–9. MegaWizard Plug-In Manager [page 1] to Edit Existing Custom Megafunction Variation**



- Select **Edit an existing custom megafunction variation**. Click **Next**.

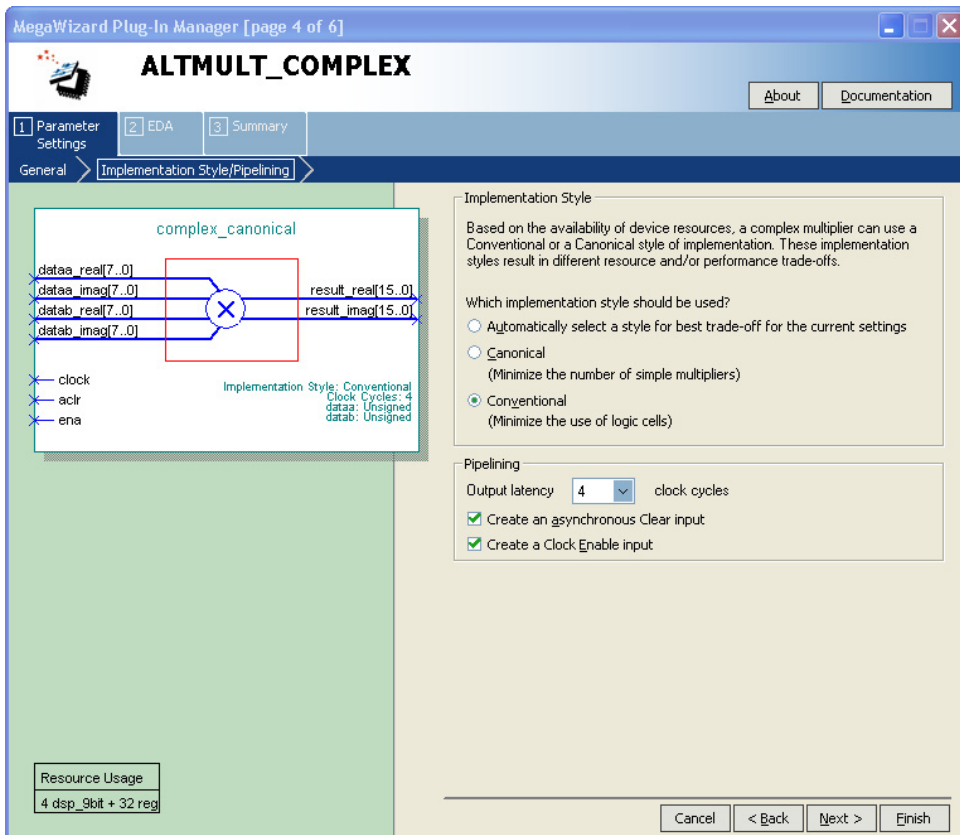
4. Browse to your working directory and select the `complex_canonical.v` file (Figure 2–10). Click **Next**.

**Figure 2–10. MegaWizard Plug-In Manager [page 2b]**



5. On page 3 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager, click **Next**.
6. On page 4 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager, select **Conventional (Minimize the use of logic cells)**, as shown in Figure 2–11. Click **Finish**.

Figure 2–11. Selecting Conventional Representation as Implementation Style



7. On page 6 of the ALTMULT\_COMPLEX MegaWizard Plug-In Manager, click **Finish**.
8. The **MegaWizard Plug-In Manager** dialog box appears. Click **OK** to overwrite the file specified.
9. The **Megafunction File** dialog box appears with an option to add the Quartus II IP Advisor File (.qip) your project. Click **Yes**.
10. On the Processing menu, click **Start Compilation**. When the **Full compilation was successful** message box appears, click **OK**.

- At the Compilation Report window, expand the Fitter menu hierarchy, followed by the Resource Section menu hierarchy. Select **DSP Block Usage Summary**. The number of DSP blocks used is displayed (Figure 2–12).

**Figure 2–12. Resource Usage for Conventional Representation**

Fitter DSP Block Usage Summary				
	Statistic	Number Used	Available per Block	Maximum Available
1	Simple Multipliers (9-bit)	0	8	96
2	Simple Multipliers (18-bit)	0	4	48
3	Simple Multipliers (36-bit)	0	1	12
4	Multiply Accumulators (18-bit)	0	2	24
5	Two-Multipliers Adders (9-bit)	2	4	48
6	Two-Multipliers Adders (18-bit)	0	2	24
7	Four-Multipliers Adders (9-bit)	0	2	24
8	Four-Multipliers Adders (18-bit)	0	1	12
9	Dynamic DSP Blocks	0	1	12
10	DSP Blocks	1	--	12
11	DSP Block 9-bit Elements	4	8	96
12	Signed Multipliers	0	--	--
13	Unsigned Multipliers	4	--	--
14	Mixed Sign Multipliers	0	--	--
15	Variable Sign Multipliers	0	--	--
16	Dedicated Shift Register Chains	0	--	--



The resource usage values are correct at the time of publication. For updated results, refer to the Compilation Report of the latest version of the Quartus II software.

For designs with input data widths that are less than 18 bits, Altera recommends that you use the canonical representation for efficient resource usage. If you set the `IMPLEMENTATION_STYLE` parameter to `AUTO`, the MegaWizard Plug-In Manager chooses the canonical representation by default for designs using non-Stratix III devices or input widths that are less than 18 bits.

For designs using Stratix III devices or input widths of 18 bits and above, you can use only the conventional representation. If you set the `IMPLEMENTATION_STYLE` parameter to `AUTO`, the MegaWizard Plug-In Manager chooses the conventional representation by default.

## Functional Simulation in the ModelSim-Altera Simulator

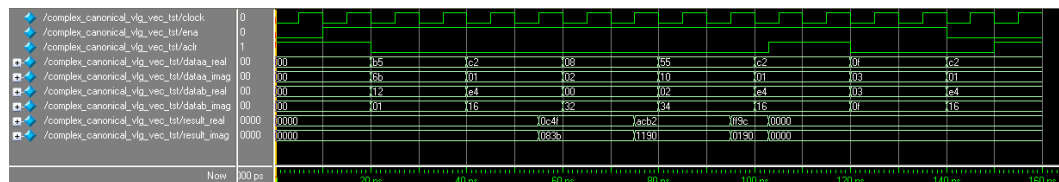
Run the ModelSim-Altera software to generate a behavior waveform of the design example.

You should be familiar with the ModelSim-Altera software before trying out the design example. If you are unfamiliar with the ModelSim-Altera software, refer to the support page for software products on the Altera website ([www.altera.com](http://www.altera.com)). On the support page, there are links to such topics as installation, usage, and troubleshooting.

Set up and simulate the design in the ModelSim-Altera software by performing the following steps:

1. Unzip the **alftp\_complex\_msim\_ex1.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files.
5. Click **OK**.
6. On the Tools menu, click **Execute Macro**.
7. Select the **complex\_canonical.do** file and click **Open**. The **complex\_canonical.do** file is a script file for the ModelSim-Altera software to automate all the necessary settings for the simulation.
8. View the simulation results in the Wave window ([Figure 2-13](#)). You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in the **complex\_canonical.do** file.

**Figure 2-13. Simulation Waveforms in the ModelSim-Altera Software**



## Conclusion

The Quartus II software provides parameterizable megafunctions ranging from simple arithmetic units, such as adders and counters, to advanced phase-locked loop (PLL) blocks, divisions, and memory structures. These megafunctions are performance-optimized for Altera devices, and therefore, provide more efficient logic synthesis and device implementation, because they automate the coding process and save valuable design time. Altera recommends using these functions during design implementation so you can consistently meet your design goals.

### Ports and Parameters

The Quartus® II software provides the Complex Multiplier (ALTMULT\_COMPLEX) megafunction that implements the multiplication of two complex numbers. This chapter describes the ports and parameters of the ALTMULT\_COMPLEX megafunction. These ports and parameters are available to customize the ALTMULT\_COMPLEX megafunction according to your application.

The parameter details are only relevant if you bypass the MegaWizard® Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in your design. The details of these parameters are hidden if you use the MegaWizard Plug-In Manager interface.



Refer to the latest version of the Quartus II Help for the most current information on the ports and parameters for this megafunction.

Figure 3–1 shows the input and output ports for the ALTMULT\_COMPLEX megafunction.

**Figure 3–1. Input and Output Ports of ALTMULT\_COMPLEX Megafunction**

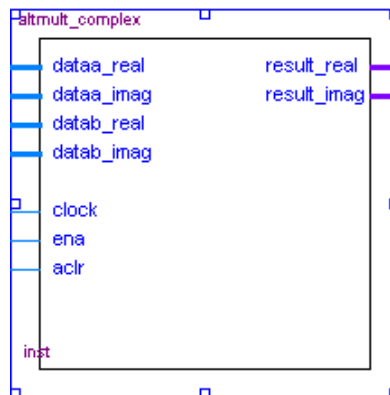


Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameters for the ALTMULT\_COMPLEX megafunction.

Port Name	Required	Description	Comments
aclr	No	Asynchronous clear for the complex multiplier.	—
clock	Yes	Clock input for the complex multiplier.	—
dataa_imag[]	Yes	Imaginary input value for the data A port of the complex multiplier.	Input port [WIDTH_A-1..0] wide. Width is specified by the WIDTH_A parameter.
dataa_real[]	Yes	Real input value for the data A port of the complex multiplier.	Input port [WIDTH_A-1..0] wide. Width is specified by the WIDTH_A parameter.
datab_imag[]	Yes	Imaginary input value for the data B port of the complex multiplier.	Input port [WIDTH_B-1..0] wide. Width is specified by the WIDTH_B parameter.
datab_real[]	Yes	Real input value for the data B port of the complex multiplier.	Input port [WIDTH_B-1..0] wide. Width is specified by the WIDTH_B parameter.
ena	No	Clock enable for the clock port of the complex multiplier.	—

Table 3–2 lists the output ports of the ALTMULT\_COMPLEX megafunction.

Port Name	Required	Description	Comments
result_imag	Yes	Imaginary output value of the multiplier.	Output port [WIDTH_RESULT-1..0] wide. Width is specified by the WIDTH_RESULT parameter.
result_real	Yes	Real output value of the multiplier.	Output port [WIDTH_RESULT-1..0] wide. Width is specified by the WIDTH_RESULT parameter.

Table 3–3 lists the parameters for the ALTMULT\_COMPLEX megafunction.

**Table 3–3. ALTMULT\_COMPLEX Megafunction Parameters (Part 1 of 2)**

Parameter Name	Type	Required	Description
IMPLEMENTATION_STYLE	String	Yes	Specifies the representation algorithm and the number of bits per channel. Values are AUTO, CANONICAL, and CONVENTIONAL. A value of AUTO directs the Quartus II software to determine the best implementation method based on the selected device family and input width. A value of CANONICAL is available for input widths that are less than 18 bits and for all supported devices, except for Stratix III devices. A value of CONVENTIONAL is available for all supported device families for the entire input range (1 to 256 bits). If omitted, the default is AUTO.
PIPELINE	Integer	Yes	Specifies the amount of latency, in clock cycles, needed to produce the result. Values are [0 . . 14]. If the value of IMPLEMENTATION_STYLE is CANONICAL, the maximum value of PIPELINE is 14. If the value of IMPLEMENTATION_STYLE is CONVENTIONAL, the maximum value of PIPELINE is 11. If omitted, the default is 4.
REPRESENTATION_A	String	Yes	Specifies the number representation of data A. Values are UNSIGNED and SIGNED. The data A inputs are interpreted as unsigned numbers when the value is UNSIGNED, and as signed two's-complement numbers when the value is SIGNED. If omitted, the default is UNSIGNED.
REPRESENTATION_B	String	Yes	Specifies the number representation of data B. Values are UNSIGNED and SIGNED. The data B inputs are interpreted as unsigned numbers when the value is UNSIGNED, and as signed two's-complement numbers when the value is SIGNED. If omitted, the default is UNSIGNED.
WIDTH_A	Integer	Yes	Specifies the width of the dataa_real[] and dataa_imag[] ports. Value must be 256 bits or less. If omitted, the default is 18 bits.
WIDTH_B	Integer	Yes	Specifies the width of the datab_real[] and datab_imag[] ports. Value must be 256 bits or less. If omitted, the default is 18 bits.
WIDTH_RESULT	Integer	Yes	Specifies the width of the result_real[] and result_imag[] ports. Value must be 256 bits or less. If omitted, the default is 36 bits.

**Table 3–3. ALTMULT\_COMPLEX Megafunction Parameters (Part 2 of 2)**

Parameter Name	Type	Required	Description
INTENDED_DEVICE_FAMILY	String	No	This parameter is used for modeling and behavioral simulation purposes. Create the ALTMULT_COMPLEX megafunction with the MegaWizard Plug-in Manager to calculate the value for this parameter.
LPM_HINT	String	No	Allows you to specify Altera-specific parameters in the VHDL Design File (.vhd). The default is UNUSED.
LPM_TYPE	String	No	Identifies the entity name of the library of parameterized modules (LPM) in .vhd files.