



Memory-Based Multiplier (ALTMEMMULT)

Megafunction User Guide



101 Innovation Drive
San Jose, CA 95134
www.altera.com

Software Version:	8.0
Document Version:	3.0
Document Date:	July 2008

Copyright © 2008 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the 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-MF91004-3.0

Chapter 1. About this Megafunction

Device Family Support	1-1
Introduction	1-1
Features	1-1
General Description	1-1
Common Applications	1-2
Resource Utilization and Performance	1-3

Chapter 2. Getting Started

System and Software Requirements	2-1
MegaWizard Plug-In Manager Customization	2-1
MegaWizard Plug-In Manager Page Descriptions	2-1
Instantiating Megafunctions in HDL Code or Schematic Designs	2-6
Generating a Netlist for EDA Tool Use	2-7
Using the Port and Parameter Definitions	2-7
Identifying a Megafunction after Compilation	2-8
Simulation	2-8
Quartus II Software Simulation	2-8
EDA Tool Simulation	2-9
Design Example: 8 x 8 Multiplier	2-9
Design Files	2-10
Functional Simulation in the Quartus II Software	2-11
Functional Simulation in the ModelSim-Altera Simulator	2-12
Understanding the Simulation Results	2-13
Conclusion	2-16

Chapter 3. Specifications

Ports and Parameters	3-1
----------------------------	-----

Additional Information

Revision History	Info-1
Referenced Documents	Info-2
How to Contact Altera	Info-2
Typographic Conventions	Info-2

Device Family Support

The ALTMEMMULT megafunction supports the following target Altera® device families:

- Arria® GX
- Cyclone® III
- Cyclone II
- Cyclone
- HardCopy® II
- HardCopy Stratix®
- Stratix IV
- Stratix III
- Stratix II GX
- Stratix II
- 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 ALTMEMMULT megafunction implements a multiplier and offers many additional features, which include the following:

- Support for both signed and unsigned data representation formats
- Options for implementation in memory blocks
- Support for pipelining with parameterized output latency
- Synchronous-clear and load-control inputs
- Validation of results

General Description

The ALTMEMMULT megafunction is one of the arithmetic megafunctions provided in the Quartus® II MegaWizard® Plug-In Manager. The ALTMEMMULT megafunction lets you create a memory-based multiplier using the on-chip memory blocks found in

Altera FPGAs (with M512, M4K, M9K, and MLAB memory blocks), based on distributed arithmetic calculations. Distributed arithmetic calculations perform multiplication by distributing the operation over many look-up tables (LUTs) and shift-accumulating the results.

The ALTMEMMULT megafunction supports both signed and unsigned multiplication. The ALTMEMMULT megafunction lets you specify the throughput as the number of clock cycles required to complete each multiplication. As few as one partial product look-up RAM can be used, which minimizes the resources required to implement the multiplier, at the cost of reduced throughput. For example, using a single M512 block and approximately 50 logic elements (LEs), the ALTMEMMULT megafunction can create a multiplier capable of calculating a new 15×13 multiplication every 3 clock cycles with a latency of 6 clock cycles.

The ALTMEMMULT megafunction is a synchronous function that requires a clock. The ALTMEMMULT megafunction and the MegaWizard Plug-In Manager create a multiplier with the smallest throughput and latency possible for a given set of parameters and specifications.

An additional feature of the ALTMEMMULT megafunction is that it can store multiple constants in the random-access memory (RAM). This is especially useful for larger RAM blocks.

The ALTMEMMULT megafunction is used to create only memory-based multipliers. This is useful if you do not have sufficient resources to implement the multipliers in LEs or dedicated multiplier resources.



Refer to the following megafunctions for other multiplier implementations:

- [Multiplier-Adder Megafunction User Guide \(ALTMULT_ADD\)](#)
- [Multiplier Megafunction User Guide \(LPM_MULT\)](#)
- [Multiplier-Accumulator Megafunction User Guide \(ALTMULT_ACCUM\)](#)

Common Applications

Common multiplier applications include finite impulse response (FIR) filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT), digital mixers, and any other designs that require multiplication of sets of data.



For more information about implementing multipliers in Altera FPGAs, refer to *AN 306: Implementing Multipliers in FPGA Devices*.

Resource Utilization and Performance

In Stratix series and Cyclone series devices, the multipliers created using the ALTMEMMULT megafunction are placed in dedicated memory blocks. These multipliers are implemented as LUTs in memory, which contain all possible partial results from multiplication.

Table 1–1 summarizes the resource usage of the ALTMEMMULT megafunction when implementing a multiplier with the default settings in the MegaWizard Plug-In Manager.

Device Family	Optimization (2)	Size (3)	Output Latency (Clock Cycles) (3)	RAM Block Type (3)	Logic Usage		
					Memory Blocks	Adaptive Look-Up Tables (ALUTs)	Logic Elements (LEs)
Stratix III	Balanced	4x8	5	MLAB	1	13	39
		8x16	5	MLAB	1	25	67
		16x16	7	MLAB	1	25	87
Cyclone III	Balanced	4x8	2	M9K	1	—	—
		8x16	2	M9K	1	—	—
		16x16	5	M9K	1	33	59

Notes to Table 1–1:

- (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) Choose a design implementation that balances high performance with minimal logic usage. This setting is available for all devices listed in “Device Family Support” on page 1–1. The balanced optimization logic option is set in **Analysis and Synthesis Settings** on the Assignments menu.
- (3) Specify the settings for the memory multiplier using the MegaWizard Plug-In Manager.

System and Software 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

MegaWizard Plug-In Manager Customization

Use the MegaWizard® Plug-In Manager to create or modify design files containing custom megafunction variations that can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard that lets you specify options for the Memory-Based Multiplier (ALTMEMMULT) megafunction.

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:

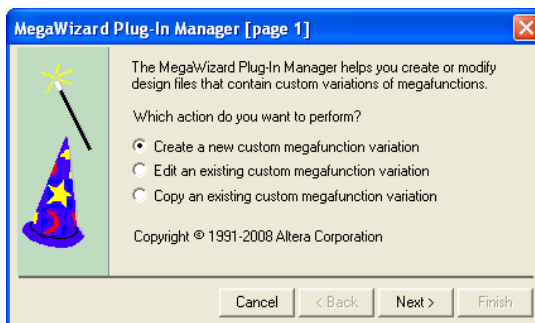
```
qmegawiz ←
```

MegaWizard Plug-In Manager Page Descriptions

This section provides descriptions of the options available on the individual pages of the ALTMEMMULT wizard. Click **Next** on each page to move to the next page.

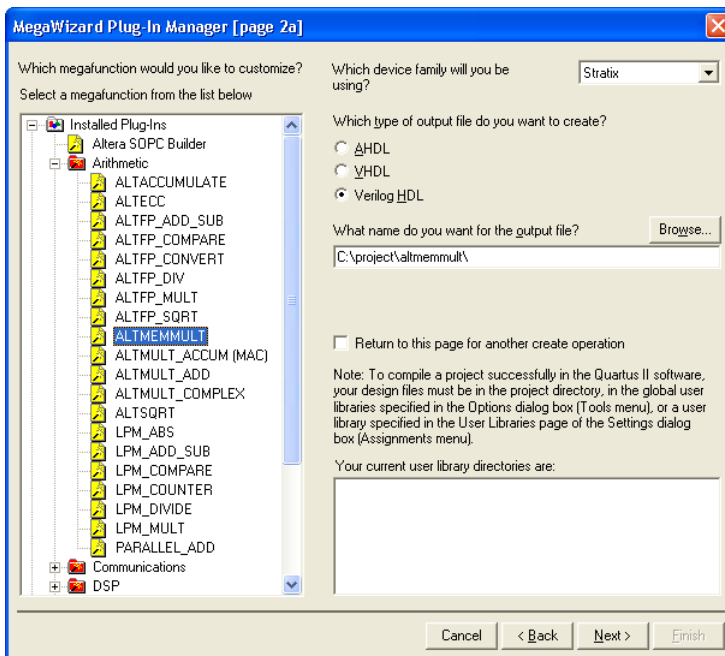
On page 1 of the MegaWizard Plug-In Manager, you can select **Create a new custom megafunction variation**, **Edit an existing custom megafunction variation**, or **Copy an existing custom megafunction variation** (Figure 2-1).

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. The ALTMEMMULT megafunction appears under the Arithmetic category.

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



On page 3 of the ALTMEMMULT MegaWizard Plug-In Manager, specify the widths for the input bus and coefficient, specify the value of the initial coefficient, specify the representations of the input bus and coefficient, select additional ports, and specify the RAM (random-access memory) block type (Figure 2–3).

Figure 2–3. ALTMEMMULT MegaWizard Plug-In Manager [page 3 of 5]

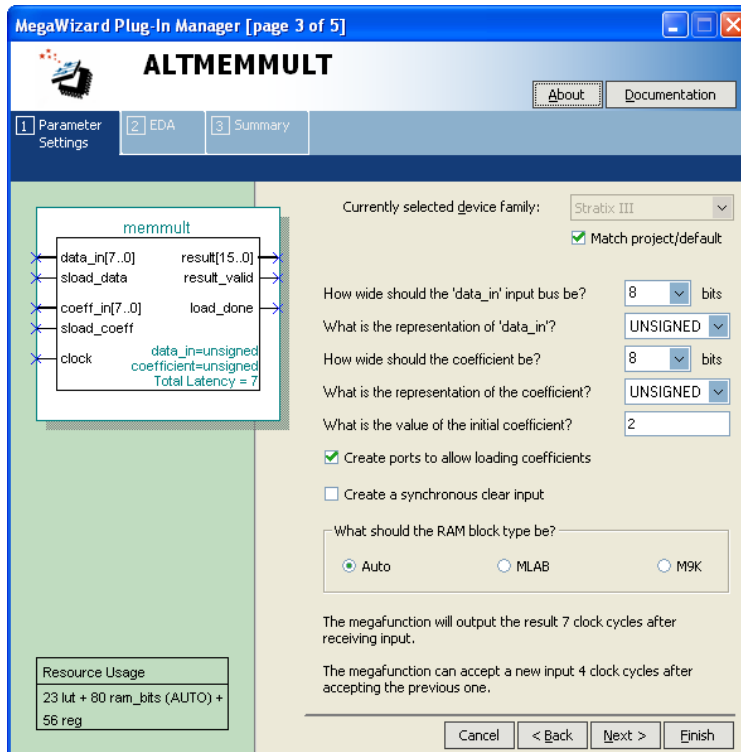
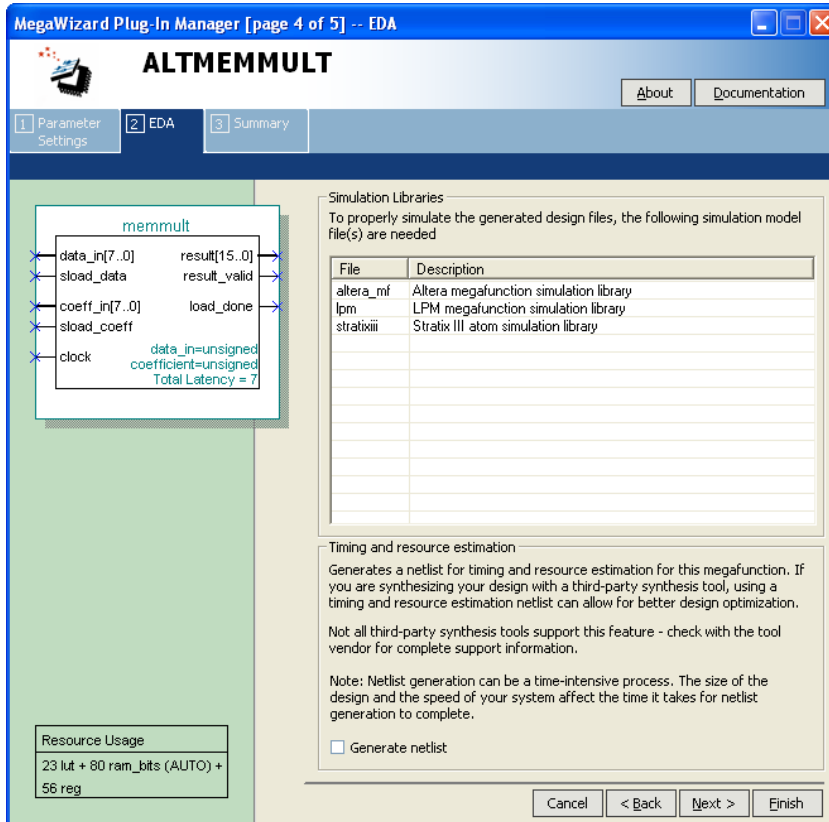


Table 2–1 shows the options available on page 3 of the ALTMEMMULT MegaWizard Plug-In Manager.

Table 2–1. ALTMEMMULT MegaWizard Plug-In Manager [page 3] Options	
Configuration Setting	Description
Currently selected device family	Shows the device family you selected on page 2a of the MegaWizard Plug-In Manager. Turn off Match project/default to change the device family.
How wide should the 'data_in' input bus be?	Select the width of the input bus from 2 to 32 bits.
What is the representation of 'data_in'?	Specify SIGNED or UNSIGNED for the input bus.
How wide should the coefficient be?	Specify the width of the coefficient.
What is the representation of the coefficient?	Specify SIGNED or UNSIGNED for the coefficient.
What is the value of the initial coefficient?	Specify the initial value of the coefficient.
Create ports to allow loading coefficients	Turn on this option to create additional ports to load coefficients. If turned on, the <code>coeff_in[x..o]</code> port is created. You can vary the coefficients when necessary.
Create a synchronous clear input	Turn on this option to create a synchronous-clear input to clear the output ports at the rising edge of clock.
What should the RAM block type be?	Select the RAM block type.

On page 4 of the ALTMEMMULT MegaWizard Plug-In Manager, you can choose to generate a resource usage and timing estimation netlist (Figure 2–4).

Figure 2–4. ALTMEMMULT MegaWizard Plug-In Manager [page 4 of 5]

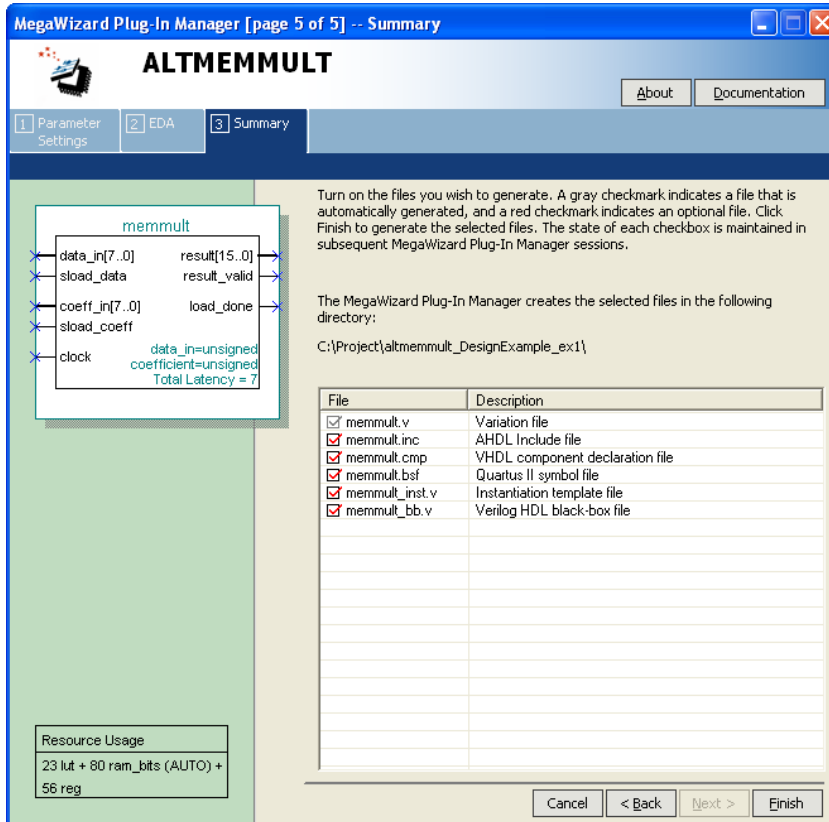


On page 5 of the ALTMEMMULT MegaWizard Plug-In Manager, specify the types of files to be generated. You can choose from the following types of files:

- 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 file (<function name>_bb.v)

If you selected **Generate netlist** on page 4, 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 2-5).

Figure 2-5. *ALTMEMMULT MegaWizard Plug-In Manager [page 5 of 5]*



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 the 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` and `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: 8 x 8 Multiplier

This section presents a design example that shows how the ALTMEMMULT megafunction generates a basic multiplier. The objective of this design example is to determine the 16-bit product of two unsigned 8-bit numbers.

This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described in detail. When you are finished with this example, you can incorporate it into an overall design.

In this example, you perform the following tasks:

1. Generate an 8-bit multiplier using the ALTMEMMULT megafunction and the MegaWizard Plug-In Manager
2. Implement the multiplier function in the device by assigning the EP2S30F484C3 device to the project and compiling the project
3. Simulate the 8-bit multiplier design module

Design Files

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

The following functional simulations are available for the design example:

- [“Functional Simulation in the Quartus II Software” on page 2–11](#)
- [“Functional Simulation in the ModelSim-Altera Simulator” on page 2–12](#)

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

Configuration Settings

This design example configures the ALTMEMMULT megafunction for an 8-bit multiplier.

[Table 2–2](#) shows the ALTMEMMULT 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 2–2](#). Click **Next** to advance from one page to the next..

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Select a megafunction	Under Arithmetic, select ALTMEMMULT
	Which device family will you be using?	Stratix III
	Which type of output file do you want to create?	Verilog HDL
	What name do you want for the output file?	memmult

Table 2–2. Design Example: Configuration Settings (Part 2 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
3	Currently selected family	Stratix III
	Match project/default	Selected
	How wide should the 'data_in' input bus be?	8
	What is the representation of 'data_in'?	Unsigned
	How wide should the coefficient be?	8
	What is the representation of the coefficient?	Unsigned
	What is the value of the initial coefficient?	2
	Create ports to allow loading coefficients	Selected
	Create a synchronous clear input	Not selected
	What should the RAM block type be?	Auto
4	Generate netlist	Not selected
5	Variation file	Selected
	Instantiation template file	Not selected
	Verilog HDL black-box file	Not selected
	AHDL Include file	Not selected
	VHDL component declaration file	Not selected
	Quartus II symbol file	Not selected
	Quartus II IP Advisor file	Not selected

Functional Simulation in the Quartus II Software

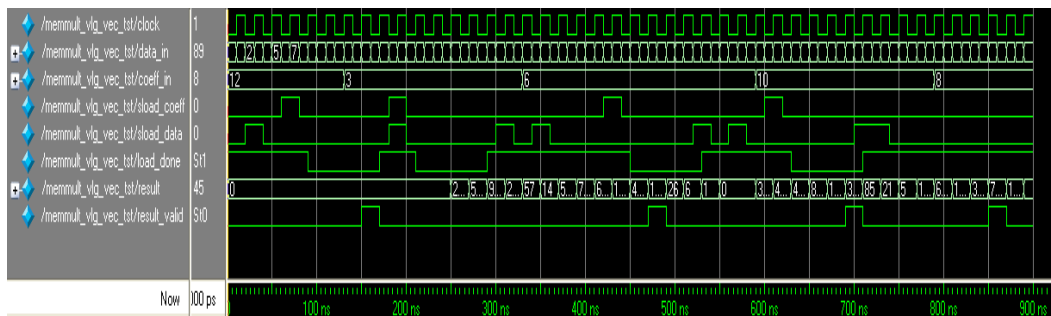
Run the Quartus II functional simulation to display the functional behavior waveform of the design example. The design files are already configured and compiled.

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

1. Open the **altmemmult_DesignExample_ex1.zip** project and extract the **memmult.qar** file.
2. In the Quartus II software, open the **memmult.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**.

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 **memmult.do** file and click **Open**. The **memmult.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 2–7). You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in the **memmult.do** file.

Figure 2–7. Simulation Waveforms in the ModelSim-Altera Software



Understanding the Simulation Results

The ALTMEMMULT megafunction multiplies the value of the `data_in` input port and the current coefficient value stored in the memory to produce:

$\text{result}(x,y)$

where:

- x is the value of `data_in` port
- y is the current value of the `coeff_in` port

If the value of the `MAX_CLOCK_CYCLES_PER_RESULT` parameter is more than 1, the `sload_data` signal indicates a new multiplication and the `result_valid` signal indicates the validity of the multiplication result. If the value of the `MAX_CLOCK_CYCLES_PER_RESULT` parameter is 1, the `sload_data` signal is not used and every positive clock edge starts a new multiplication.

In this design example, with the `MAX_CLOCK_CYCLES_PER_RESULT` parameter set to 4, the design requires no less than 4 clock cycles to compute the multiplication. The `sload_data` signal is used to indicate a new multiplication.



Altera recommends that you do not pull the `sload_data` signal high during the 4 clock cycles when the multiplication is taking place to avoid getting unpredictable results.

The megafunction can only receive new inputs after 4 clock cycles. With the `TOTAL_LATENCY` parameter set to 7, all multiplication results require 7 clock cycles to appear at the `result` port.

The `COEFFICIENT0` parameter holds the value of the first fixed coefficient, which is set to 2 (`COEFFICIENT0=2`) for this design example. The megafunction uses the latest coefficient value for every multiplication.

The `sload_data` signal asserts when a new coefficient value is written into the register. The `load_data` signal pulls low 1 clock cycle after the `sload_data` signal deasserts. When the `load_data` signal is low, the new coefficient value is reprogrammed into the RAM look-up table. Until the `load_data` signal pulls high, no other coefficient value can be loaded into the memory (regardless of whether the `sload_data` signal asserts anytime in between). The `load_data` signal asserts when programming completes.

The load time required to write a new coefficient value into the register is the same for any instance of the `ALTMEMMULT` megafunction. However, it can vary depending on the size of the RAM used.

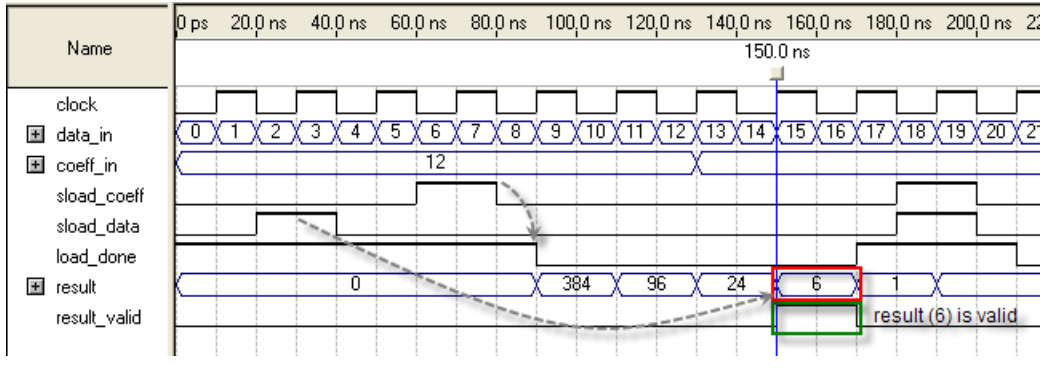
Figure 2–8. Multiplication with Coefficient of 2

Figure 2–8 shows the following:

- At 30.0 ns, the `sload_data` signal asserts and triggers the first multiplication between the `data_in` value of 3 and the `COEFFICIENT0` value of 2. The result is sent to the `result` port 7 clock cycles later at 150.0 ns. The `result_valid` port asserts to indicate that the multiplication is valid.
- At 70.0 ns, the `sload_coeff` signal asserts to register a new coefficient value of 12 into the register. The `load_done` signal pulls low to begin loading the new value into the memory, and pulls high at 170.0 ns when loading is complete. In this example, the load time is 5 clock cycles.

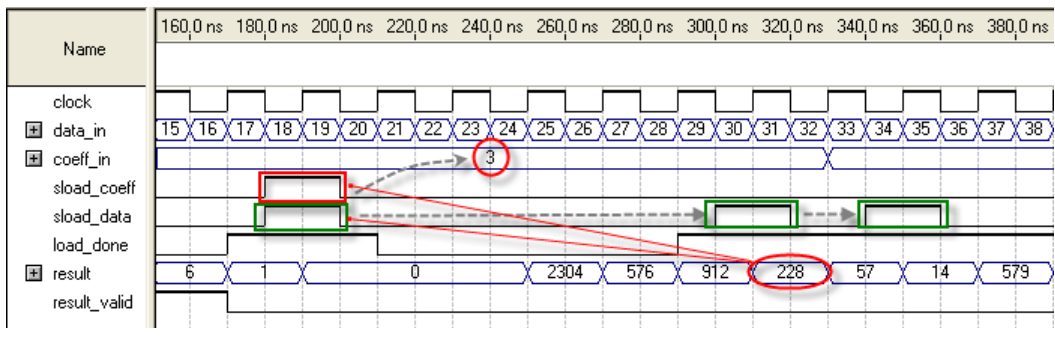
Figure 2–9. Multiplication with Coefficient of 3

Figure 2–9 shows the following:

- At 190.0 ns, the `sload_coeff` signal asserts to register a new coefficient value of 3.
- At 190.0 ns, the `sload_data` signal asserts and triggers a new multiplication. The latest value of the coefficient loaded into the memory is 12. Multiplication occurs between the `data_in` value of 19 and a coefficient of 12.
- At the same time, at 190.0 ns, the `sload_coeff` signal asserts and triggers the programming of coefficient 3. Although the multiplication result of 228 at 310.0 ns is valid, the `result_valid` signal does not pull high.



Altera recommends that you do not assert both the `sload_coeff` and `sload_data` signals at the same time to prevent the programming and computation processes from occurring simultaneously.

- At 310.0 ns, the `sload_data` signal asserts and triggers a new multiplication. However, at 350.0 ns (less than 4 clock cycles after 310.0 ns), the `sload_data` signal pulls high again and cancels the previous multiplication process. The valid result, 105, of the computation is displayed at 470.0 ns.

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, multipliers, 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 Memory-Based Multiplier (ALTMEMMULT) megafunction that lets you create a memory-based multiplier using the on-chip memory blocks found in Altera® FPGAs (with M512, M4K, M9K, and MLAB memory blocks), based on distributed arithmetic calculations. The ALTMEMMULT megafunction supports both signed and unsigned multiplication. This chapter describes the ports and parameters of the ALTMEMMULT megafunction. These ports and parameters are available to customize the ALTMEMMULT megafunction according to your application.

The parameter details are only relevant for users who use the megafunction as a directly parameterized instantiation in their design.



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

Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameterized megafunction interface.

Table 3–1. ALTMEMMULT Megafunction Input Ports			
Port Name	Required	Description	Comments
clock	Yes	Clock input port for the multiplier.	Input port wide.
coeff_in[]	No	Coefficient input port for the multiplier.	Input port [WIDTH_C–1..0] wide.
data_in[]	Yes	Data input port to the multiplier.	Input port [WIDTH_D–1..0] wide.
sclr	No	Synchronous clear input.	Input port [WIDTH_S–1..0] wide. If only one coefficient exists, the sel[] input port is not available.
sel[]	No	Fixed coefficient selection.	—
sload_data	No	Signal that specifies new multiplication operation and cancels any existing multiplication operation.	If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the sload_data input port is ignored.
sload_coeff	No	Replaces the current selected coefficient value with the value specified in the coeff_in input port.	—

Table 3–2. ALTMEMMULT Megafunction Output Ports

Port Name	Required	Description	Comments
result []	Yes	Multiplier output port.	Output port [WIDTH_R-1..0] wide.
result_valid	Yes	Indicates when the output is the valid result of a complete multiplication.	If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the result_valid output port is not used.
load_done	No	Indicates when the new coefficient is loaded.	—

Table 3–3. ALTMEMMULT Megafunction Parameters (Part 1 of 2)

Parameter	Type	Required	Comments
WIDTH_D	Integer	Yes	Specifies the width of the data_in[] port.
WIDTH_C	Integer	Yes	Specifies the width of the coeff_in[] port.
NUMBER_OF_COEFFICIENTS	Integer	No	Specifies the number of coefficients that are stored in the look-up table.
WIDTH_S	Integer	No	Specifies the width, in bits, of the sel[] port.
WIDTH_R	Integer	Yes	Specifies the width, in bits, of the result[] port.
COEFFICIENT0	Integer	Yes	Specifies value of the first fixed coefficient.
COEFFICIENT1..COEFFICIENTn	Integer	Yes	Specifies additional coefficients in the look-up table.
TOTAL_LATENCY	Integer	Yes	Specifies the total number of clock cycles from the start of a multiplication to the time the result is available at the result output port.
MAX_CLOCK_CYCLES_PER_RESULT	Integer	No	Specifies the number of clock cycles per result.
DATA_REPRESENTATION	String	No	Specifies whether the data_in[] input port is signed or unsigned. The valid values are SIGNED and UNSIGNED. If omitted, the default value is SIGNED.

Table 3–3. ALTMEMMULT Megafunction Parameters (Part 2 of 2)

Parameter	Type	Required	Comments
COEFF_REPRESENTATION	String	No	Specifies whether the <code>coeff_in[]</code> input port and the preloaded coefficients are signed or unsigned. The valid values are SIGNED and UNSIGNED. If omitted, the default value is SIGNED.
RAM_BLOCK_TYPE	String	No	Specifies the RAM block type. Values are AUTO, SMALL, MEDIUM, M512, M4K, M9K, and MLAB. If omitted, the default is AUTO. Both M9K and MLAB can be used for Stratix III devices and above.



Revision History The table below displays the revision history for the chapters in this User Guide.

Date and Document Version	Changes Made	Summary of Changes
July 2008 v3.0	<ul style="list-style-type: none">• Added support for Stratix® IV and Arria® GX devices• Replaced “Using the MegaWizard Plug-In Manager” section with “MegaWizard Plug-In Manager Page Descriptions” section• Updated description and screenshots in “MegaWizard Plug-In Manager Page Descriptions” section• Updated “Instantiating Megafunctions in HDL Code or Schematic Designs” section• Updated “Identifying a Megafunction after Compilation” section• Updated “Simulation” section• Updated and reorganized content, and replaced screenshots in “Design Example: 8 x 8 Multiplier” section• Added new sections on simulation in the Quartus II software and Model-Sim Altera software: “Functional Simulation in the Quartus II Software”, “Functional Simulation in the ModelSim-Altera Simulator”• Added “Understanding the Simulation Results” section	Updated document to support the Quartus® II software version 8.0 and address ADoQS issues
March 2007 v2.2	Minor update to include Cyclone® III device information; no new screenshots were added	—
December 2006 v2.1	Minor update to include Stratix III device information; no new screenshots were added	—
May 2006 v2.0	Updated for the Quartus II software version 6.0, including updated GUI and project examples, and new ModelSim®-Altera simulation section	—
December 2005 v1.1	Updated for the Quartus II software version 5.1	—
September 2004 v1.0	Initial release	—

Referenced Documents

This user guide references the following documents:

- *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*
- *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	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General) (Software Licensing)	Email	nacomp@altera.com
	Email	authorization@altera.com








Note to table:

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

Typographic Conventions

This document uses the typographic conventions shown below..

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

Visual Cue	Meaning
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: <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.

