



lpm_mult Megafunction

User Guide



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

Software Version:	7.0
Document Version:	2.2
Document Date:	March 2007

Copyright © 2006 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.



UG-MF91104-2.2



About this User Guide	v
Revision History	v
How to Contact Altera	v
Typographic Conventions	vi

Chapter 1. About this Megafunction

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

Chapter 2. Getting Started

System & Software Requirements	2-1
MegaWizard Plug-In Manager Customization	2-1
Using the MegaWizard Plug-In Manager	2-1
Inferring Megafunctions from HDL Code	2-8
Instantiating Megafunctions in HDL Code	2-8
Identifying a Megafunction after Compilation	2-8
Simulation	2-8
Quartus II Simulation	2-8
EDA Simulation	2-9
SignalTap II Embedded Logic Analyzer	2-9
Design Example: 9-Bit Multiplier	2-10
Design Files	2-10
Example	2-10
Generate a 9-Bit Multiplier	2-11
Implement the 9-Bit Multiplier	2-17
Functional Results—Simulate the 9-Bit Multiplier Design in Quartus	2-17
Functional Results—Simulate the 9-Bit Multiplier Design in ModelSim-Altera	2-18
Conclusion	2-20

Chapter 3. Specifications

Ports & Parameters	3-1
--------------------	-----



About this User Guide

Revision History

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

Date	Document Version	Changes Made
March 2007	2.2	Added Cyclone® III information. No new screenshots were taken.
December 2006	2.1	Added Stratix® III information. No new screenshots were taken.
June 2006	2.0	Updated for Quartus II 6.0 software. Also added ModelSim simulation section.
September 2004	1.0	Initial release.

How to Contact Altera

For the most up-to-date information about Altera® products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.








Information Type	USA & Canada
Technical support	www.altera.com/mysupport/ (800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)
Product literature	www.altera.com
Altera literature services	lit_req@altera.com (1)
FTP site	ftp.altera.com

Note to table:

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

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: Save As dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: f_{MAX} , lqdesigns 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: <file name>, <project name>.pof file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix n, e.g., resetn. Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\qdesigns\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets are used in a list of items when the sequence of the items is not important.
	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information about a particular topic.

Device Family Support

The `lpm_mult` megafunction supports the following target Altera® device families:

- Stratix® III
- Stratix II GX
- Stratix II
- Stratix
- Stratix GX
- Cyclone® III
- Cyclone II
- Cyclone
- HardCopy® II
- HardCopy Stratix
- MAX® II
- MAX 3000A
- MAX 7000AE
- MAX 7000 B
- MAX 7000 S
- ACEX® 1K
- APEX™ II
- APEX 20KC
- APEX 20KE
- FLEX® 10K
- FLEX 10KA
- FLEX 10KE
- FLEX 6000

Introduction

As design complexities increase, 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 megafunction's size by setting parameters.

Features

The `lpm_mult` megafunction implements the basic multiplier and offers many additional features, which include:

- Parameterizable input data widths
- Availability of an extra input data port for direct addition to the multiplication result
- Parameterizable output data widths
- Ability to specify a constant value for the `dataab[]` input to optimize implementation
- Support for both signed and unsigned data representations
- Options for implementation in dedicated or general hardware
- Support for pipelining with parameterized output latency
- Active high asynchronous clear and clock enable control inputs
- Support for area versus speed trade-off



The `altmult_add` megafunction can be used to implement a multiplier with greater flexibility and complexity compared to the `lpm_mult` megafunction. Refer to the *altmult_add Megafunction User Guide* for more information.



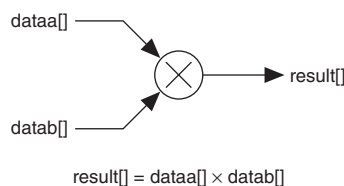
The `altmemmult` megafunction can be used to implement a multiplier using M4K or M512 memory blocks. Refer to the *altmemmult Megafunction User Guide* for more information.

General Description

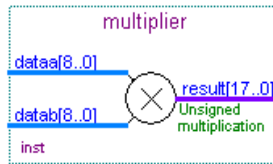
The `lpm_mult` megafunction is one of the arithmetic megafunctions provided in the Quartus® II software MegaWizard® Plug-In Manager.

The basic function of a multiplier is to multiply two input data values to produce a product as an output. [Figure 1–1](#) shows a basic multiplier.

Figure 1–1. Basic Multiplier



The `lpm_mult` megafunction is an operator that lets you multiply two sets of data. [Figure 1–2](#) shows the `lpm_mult` megafunction with input data width of nine bits and an unsigned number representation.

Figure 1–2. *lpm_mult* Megafunction Symbol

Common Applications

Multipliers are used in many common applications, such as implementing Finite Impulse Response (FIR) filters, Infinite Impulse Response (IIR) filters, Fast Fourier Transforms (FFTs), digital mixers, and any other designs that require multiplication of one or more sets of data.



For details about implementing a multiplier and using the DSP blocks and embedded multipliers in Altera devices, refer to:

- *AN 306: Techniques for Implementing Multipliers in FPGA Devices*
- The *Using the DSP Blocks in Stratix & Stratix GX Devices* chapter in volume 2 of the *Stratix Device Handbook*
- The *DSP Blocks in Stratix II Devices* chapter in volume 2 of the *Stratix II Device Handbook*
- The *Embedded Multipliers in Cyclone II Devices* chapter of the *Cyclone II Device Handbook*

Resource Utilization and Performance

The `lpm_mult` megafunction can be implemented using either logic resources or dedicated multiplier circuitry in Altera devices. Typically, the `lpm_mult` megafunction is translated to the dedicated multiplier circuitry when it is available because it provides better performance and resource utilization. If all of the input data widths are smaller than or equal to nine bits, the function uses the 9×9 multiplier configuration in the dedicated multiplier. Otherwise, 18×18 multipliers are used to process data with widths between 10 bits and 18 bits.

If you use the `sum[]` port of the `lpm_mult` megafunction, the addition is implemented using logic resources. The output adders in the Stratix, Stratix GX, and Stratix II DSP blocks cannot be used to implement the adder because the adder can only be fed by the outputs of the DSP block multipliers and not by external logic.



Refer to the DSP block and embedded multiplier chapters in the Stratix, Stratix II, and Cyclone II handbooks for information about the architecture of the DSP blocks and embedded multipliers, and for detailed information about the hardware conversion process.

Table 1–1 summarizes the resource usage for an `lpm_mult` function used to implement an 8-bit signed multiplier.

You can force the compiler to implement the multiplier in logic resources, DSP blocks, or embedded multipliers, or allow the compiler to use the default/optimum implementation.

Table 1–1. <i>lpm_mult</i> Megafunction Resource Use			
Device Family	Optimization	Width	Logic Usage
Stratix, Stratix GX	Speed	8	95 logic elements
Cyclone, Cyclone II	Balanced	8	95 logic elements
HardCopy Stratix	Area	8	95 logic elements
Stratix II	Speed	8	62 ALUTs
	Balanced	8	62 ALUTs
	Area	8	62 ALUTs



The `lpm_mult` MegaWizard Plug-In Manager reports approximate resource utilization based on the user's specification and parameters. This is reported in the bottom left corner of the MegaWizard Plug-In Manager.

System & Software Requirements

The instructions in this section require the following hardware and software:

- A PC running either Windows NT/2000/XP, Red Hat Linux 7.3 or 8.0, Red Hat Linux Enterprise 3, *or* an HP workstation running the HP-UX 11.0 operating system, *or* a Sun workstation running the Solaris 8 or 9 operating system
- Quartus® II software version 4.1 or later

MegaWizard Plug-In Manager Customization

You can use the MegaWizard® Plug-In Manager to set the `lpm_mult` megafunction features for each multiplier in the design.

Search for “`lpm_mult`” in the Quartus II Help for a listing of the parameters to use when instantiating the megafunction without using the MegaWizard Plug-In Manager.

You can start the MegaWizard Plug-In Manager in one of the following ways:

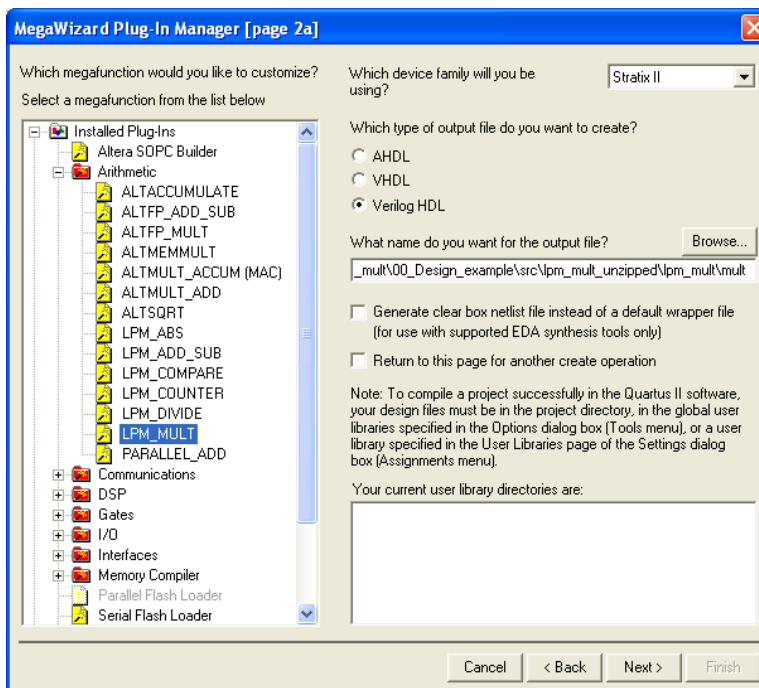
- On the Tools menu, choose **MegaWizard Plug-In Manager**.
- When working in the Block Editor, click **MegaWizard Plug-In Manager** in the Symbol window.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command on a command prompt:
`qmegawiz` ←

Using the MegaWizard Plug-In Manager

This section provides an in-depth description of each page in the `lpm_mult` megafunction. Tables 2-1, 2-2, and 2-3 show the features or settings for the `lpm_mult` megafunction. Use these tables to determine appropriate settings for your multiplier designs.

On Page 2a, select the `lpm_mult` megafunction from the Arithmetic category, select the device you intend to use, the type of output file you want to create (Verilog, VHDL, or AHDL), and what you want to name the output file. You also have the option to enable the generation of a clear-box netlist for this megafunction (Figure 2-1).

Figure 2-1. MegaWizard Plug-In Manager—`lpm_mult` [page 2a]



On Page 3 of the `lpm_mult` MegaWizard Plug-In Manager, specify the input data widths, sum input port and its data width, and the output data width (Figure 2–2).

Figure 2–2. MegaWizard Plug-In Manager—`lpm_mult` [page 3 of 7]

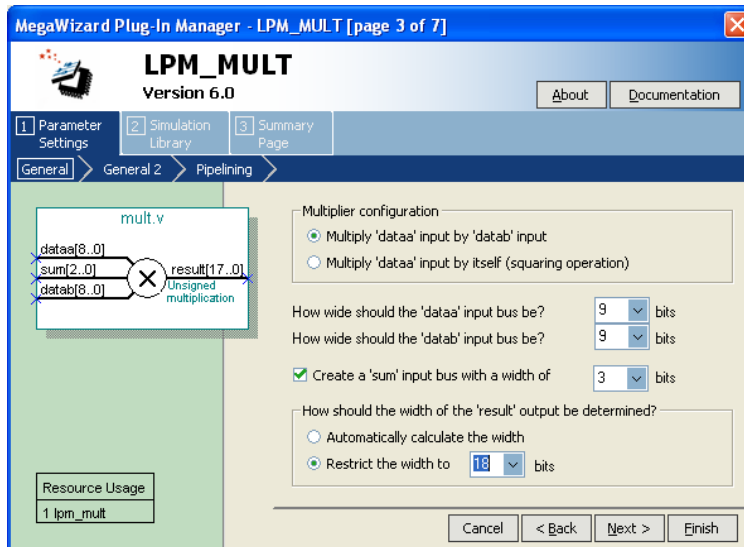


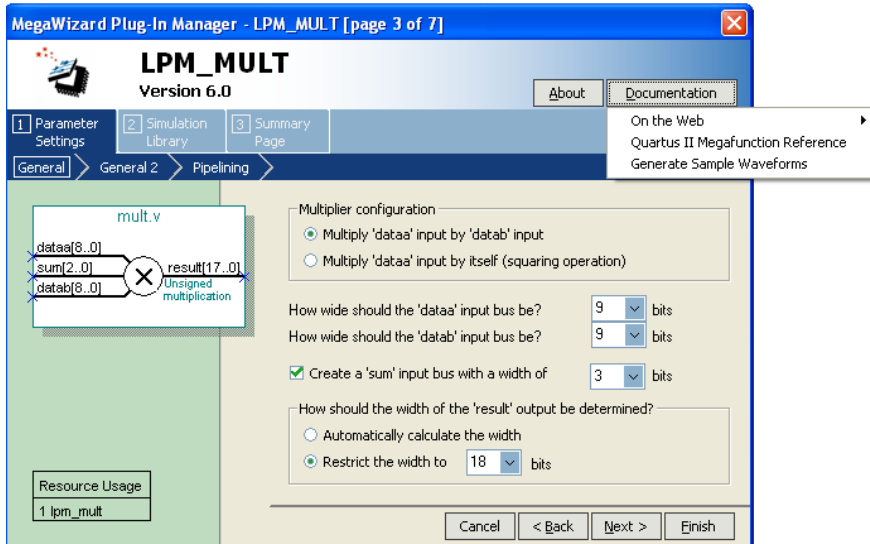
Table 2–1 shows the options available on Page 3 of the `lpm_mult` MegaWizard Plug-In Manager.

Table 2–1. `lpm_mult` MegaWizard Plug-In Manager Page 3 Options

Function	Description
Multiplier configuration	Indicate whether the <code>dataa[]</code> is to be multiplied by <code>datab[]</code> or multiplied by itself (squared)
How wide should the 'dataa' input bus be?	Specify the width of the <code>dataa[]</code> input
How wide should the 'datab' input bus be?	Specify the width of the <code>datab[]</code> input
Do you want a 'sum' input bus?	Specify a <code>sum[]</code> input port and its data width
How should the width of the 'result' output be determined?	Specify the data width for the output. This width can be automatically determined by the function based on the input bit widths, or you can specify it.

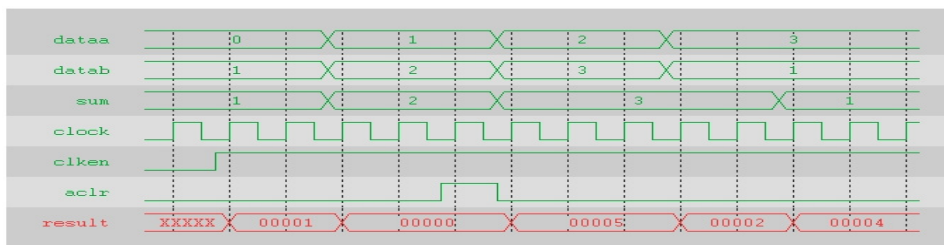
Starting on Page 3 of the `lpm_mult` MegaWizard Plug-In Manager, you can generate a sample simulation waveform, launch *AN 306: Techniques for Implementing Multipliers in FPGA Devices*, and launch the Quartus II software help for the `lpm_mult` megafunction. Select the options **On the Web**, **Quartus II Megafunction Reference**, or **Generate Sample Waveforms** from the **Documentation** button (see Figure 2–3).

Figure 2–3. `lpm_mult` Megafunction Documentation Options



These documentation options are only available in the Quartus II software version 4.1 and later.

The sample waveform illustrates the behavior of the `lpm_mult` megafunction for the chosen set of parameters in the `lpm_mult` design module (Figure 2–4). This option generates a sample waveform in HTML format in the specified `lpm_mult` design directory. The HTML file contains descriptions showing the multiplier operation.

Figure 2–4. Sample Waveforms for the *lpm_mult* Megafunction

These waveforms show the behavior of the *lpm_mult* megafunction for the chosen set of parameters. The design is an 8×8 unsigned multiplier that produces a 16-bit output. The design has a sum input of width 1.

On Page 4 of the *lpm_mult* MegaWizard Plug-In Manager, specify whether the *datab* input bus value is variable or constant, set the sign representation of the inputs, and set the hardware implementation (Figure 2–5).

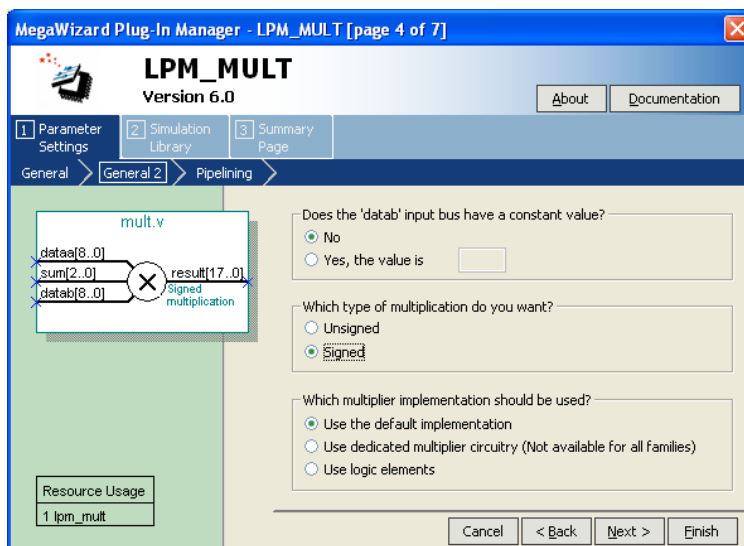
Figure 2–5. MegaWizard Plug-In Manager—*lpm_mult* [page 4 of 7]

Table 2–2 shows the options available on Page 4 of the `lpm_mult` MegaWizard Plug-In Manager.

<i>Table 2–2. lpm_mult MegaWizard Plug-In Manager Page 4 Options</i>	
Function	Description
Does the 'datab' input bus have a constant value?	Select Yes to set the width of <code>datab[]</code> port to a constant value.
Which type of multiplication do you want?	Select the sign representation of the inputs: Unsigned or Signed . The default is Unsigned . The signed representation for all library of parameterized modules (LPM) megafunctions is two's complement.
Which multiplier implementation should be used?	Select the hardware implementation type: dedicated circuitry or logic elements. The dedicated circuitry option implements the multiplier in DSP blocks or embedded multipliers and is only available in the Stratix®, Stratix II, Stratix GX, Cyclone™ II, HardCopy® Stratix, and Mercury™ device families. The default value is to use dedicated circuitry where reasonable. For example, unless otherwise specified, if the input bit width is less than five bits, the Quartus II software defaults to using logic resources.

On Page 5 of the `lpm_mult` MegaWizard Plug-In Manager, choose options and settings to control the output latency for pipelining, enabling active high asynchronous clear and clock enable inputs, and selecting the optimization technique for the multiplier function (see Figure 2–6).

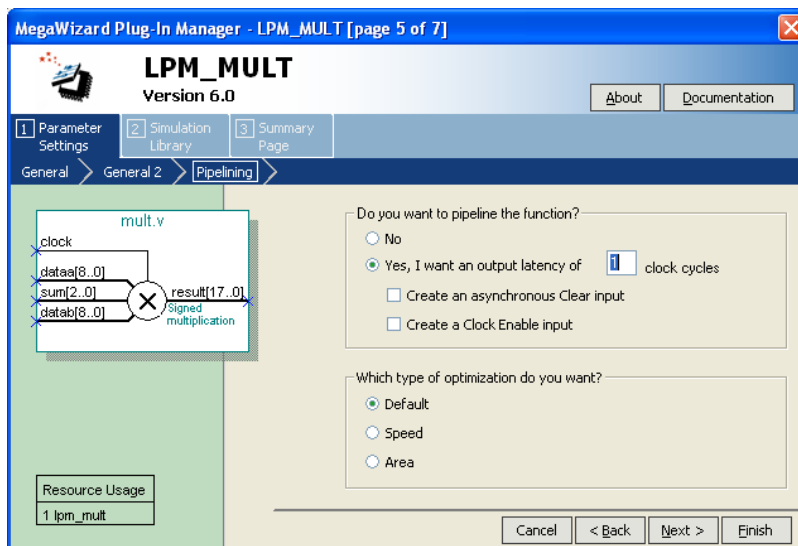
Figure 2–6. MegaWizard Plug-In Manager—lpm_mult [page 5 of 7]

Table 2–3 shows the options available on the Page 5 of the lpm_mult MegaWizard Plug-In Manager.

Table 2–3. lpm_mult MegaWizard Plug-In Manager Page 5 Options

Function	Description
Do you want to pipeline the function?	Creates a clock port to register and provides a pipelined operation for the lpm_mult function.
Create an asynchronous Clear input	Enables an active high asynchronous clear signal for registered usage.
Create a Clock Enable input	Enables an active high clock enable signal for registered usage.
Which type of optimization do you want?	Allows selection of the optimization technique for the multiplier function. The default optimization is for area.

Inferring Megafunctions from HDL Code

Synthesis tools, including Quartus II integrated synthesis, recognize certain types of HDL code and automatically infer the appropriate megafunction when a megafunction will provide optimal results. The Quartus II software uses the Altera megafunction code when compiling your design—even if you did not specifically instantiate the megafunction. The Quartus II software infers megafunctions because they are optimized for Altera devices, so the area and performance may be better than generic HDL code. Additionally, you must use megafunctions to access certain Altera architecture-specific features, such as memory, DSP blocks, and shift registers, that generally provide improved performance compared with basic logic elements.



Refer to volume 1 of the *Quartus II Handbook* for specific information about your particular megafunction.

Instantiating Megafunctions in HDL Code

When you use the MegaWizard Plug-In Manager to set up and parameterize a megafunction, it creates either a VHDL or Verilog HDL wrapper file that instantiates the megafunction (a black-box methodology). For some megafunctions, you can generate a fully synthesizable netlist for improved results with EDA synthesis tools, such as Synplify and Precision RTL Synthesis (a clear-box methodology). Both clear-box and black-box methodologies are described in volume 1 of the *Quartus II Handbook*.

Identifying a Megafunction after Compilation

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

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

Simulation

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

Quartus II Simulation

The Quartus II Simulator is a powerful tool for testing and debugging the logical operation and internal timing of Altera megafunctions instantiated in your design.

With the Quartus II Simulator, you can perform two types of simulations: functional and timing. A functional simulation in the Quartus II program enables you to verify the logical operation of your design without taking into consideration the timing delay in the FPGA. This simulation is performed using only RTL code. When performing a functional simulation, add only signals that exist before synthesis. With the registers, you can find 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, 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 might 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 and VHDL synthesis attributes that direct Analysis & 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. More information about these attributes is available in volume 1 of the *Quartus II Handbook*.

EDA Simulation

Depending on the simulation tool you are using, refer to the corresponding chapter in volume 3 of the *Quartus II Handbook*. The *Quartus II Handbook* chapters show you how to perform functional and gate-level timing simulations that include the megafunctions, with details on the files that are needed and the directories where those files are located.

SignalTap II Embedded Logic Analyzer

The SignalTap® II embedded logic analyzer provides you with a non-intrusive method of debugging all of the Altera megafunctions within your design. With the SignalTap II embedded logic analyzer, you can capture and analyze data samples for the top-level ports of the Altera megafunctions in your design while your system is running at full speed.

To monitor signals from your Altera megafunctions, first configure the SignalTap II embedded logic analyzer in the Quartus II software, and then include the analyzer as part of your Quartus II project. The Quartus II software then embeds the analyzer along with your design in the selected device.



For more information about using the SignalTap II embedded logic analyzer, refer to volume 3 in the *Quartus II Handbook*.

Design Example: 9-Bit Multiplier

Multipliers are one of the basic building blocks that are commonly used in digital signal processing (DSP) applications. For example, multipliers are often used to implement Finite Impulse Response (FIR) filters.



The `altmult_add` megafunction can be used to implement a multiplier with greater flexibility and complexity compared than the `lpm_mult` megafunction. Refer to the *altmult_add Megafunction User Guide* for more information on the `altmult_add` megafunction.



The `altmemmult` megafunction can be used to implement a multiplier using M4K or M512 memory blocks. Refer to the *altmemmult Megafunction User Guide* for more information on the `altmemmult` megafunction.

This section presents a design example that uses the `lpm_mult` megafunction to generate a basic multiplier. This example uses the MegaWizard Plug-In Manager in the Quartus II software to customize this megafunction. 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.

Design Files

The example design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site: www.altera.com.

Example

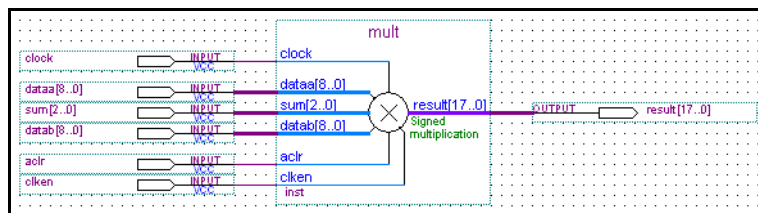
This example shows how to instantiate an `lpm_mult` megafunction using the MegaWizard Plug-In Manager. In this case, an `lpm_mult` is instantiated with all the features enabled. This example also shows simulation results that illustrate the behavior of the `lpm_mult` megafunction for the chosen set of parameters in the design. You can change the parameters as needed for your design.

In this example, you perform the following tasks:

- Generate a 9-bit multiplier.
- Implement the multiplier in architecture by assigning the Stratix II EP2S15F484C3 device and compile the project.
- Simulate the customized multiplier.

Figure 2–7 shows the `lpm_mult` megafunction design.

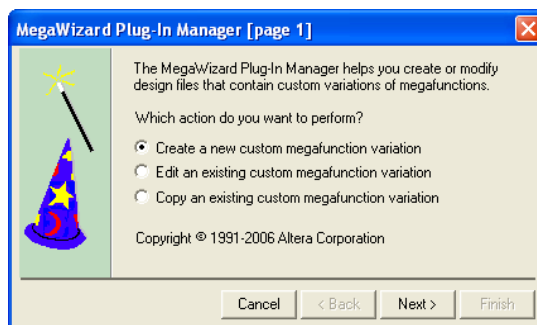
Figure 2–7. `lpm_mult` Megafunction Design



Generate a 9-Bit Multiplier

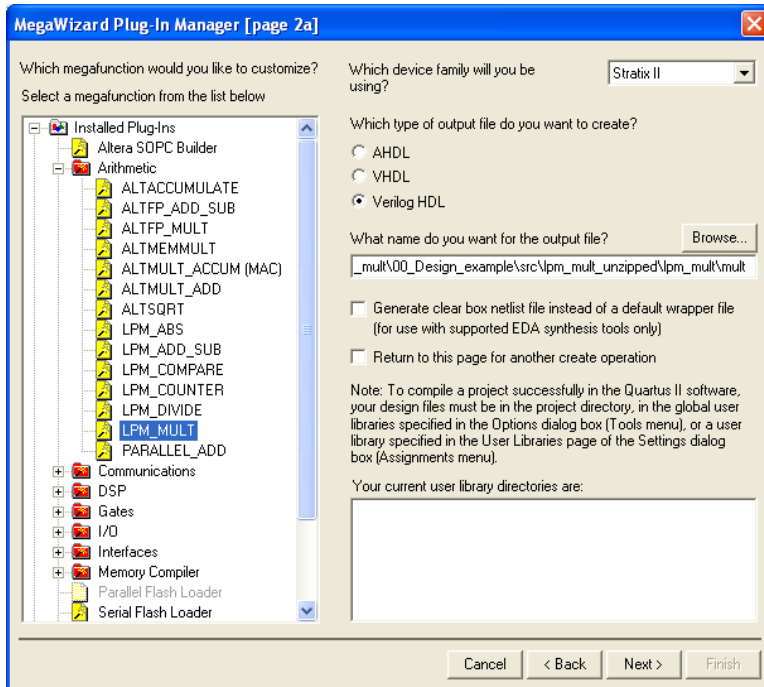
1. Open the project file **multiplier.qpf**.
2. Open the top-level file **multiplier.bdf**. This file is an incomplete file that you will complete in the course of this example.
3. Double-click on a blank area in the block design (**.bdf**) file and then click the **MegaWizard Plug-In Manager** button from the Symbol window or, on the Tools menu, choose **MegaWizard Plug-In Manager**.
4. On Page 1 of the MegaWizard Plug-In Manager, to answer the question **What action do you want to perform?**, click **Create a new custom megafunction variation** (Figure 2–8).

Figure 2–8. MegaWizard Plug-In Manager—`lpm_mult` [page 1]



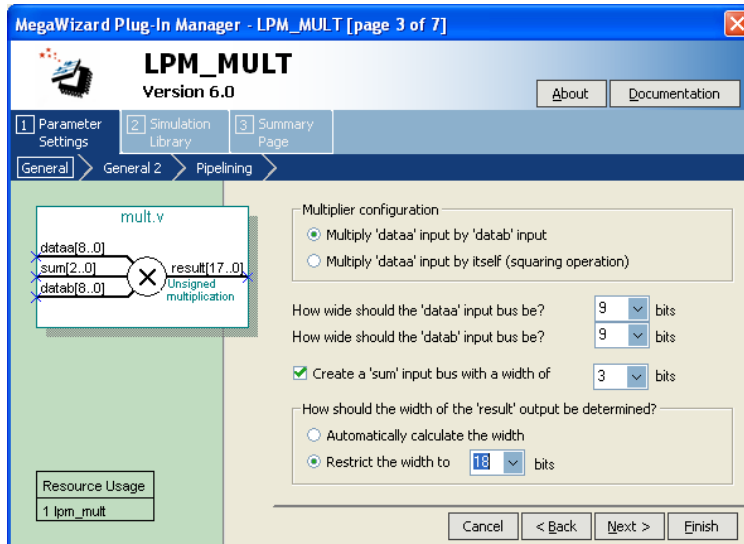
5. Click **Next**. Page 2a displays (Figure 2–9).

Figure 2–9. MegaWizard Plug-In Manager—lpm_mult [page 2a]



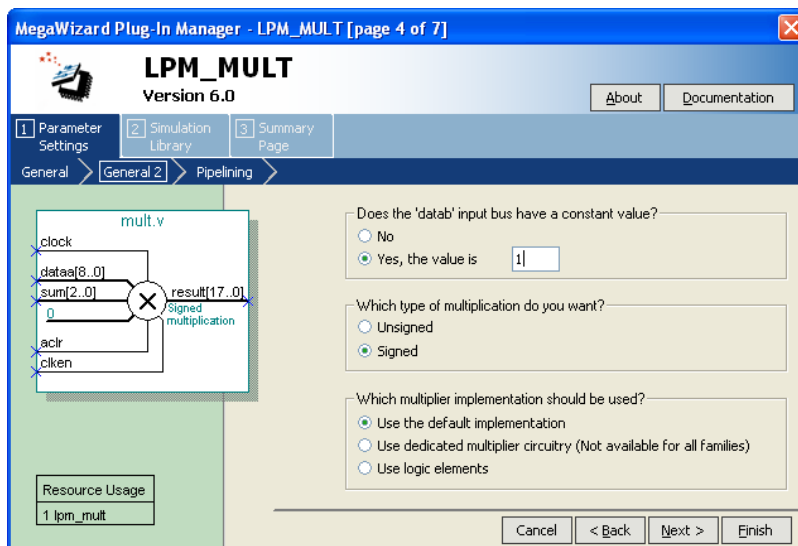
6. On page 2a, expand the **Arithmetic** folder and select the **LPM_MULT** megafunction.
7. To answer the question **Which device family will you be using?**, select **Stratix II**.
8. To answer the question **What type of output file do you want to create?**, select **Verilog HDL**.
9. Name the file `<project directory>\mult`.
10. Click **Next**. Page 3 displays (Figure 2–10).

Figure 2–10. MegaWizard Plug-In Manager—lpm_mult [page 3 of 7]

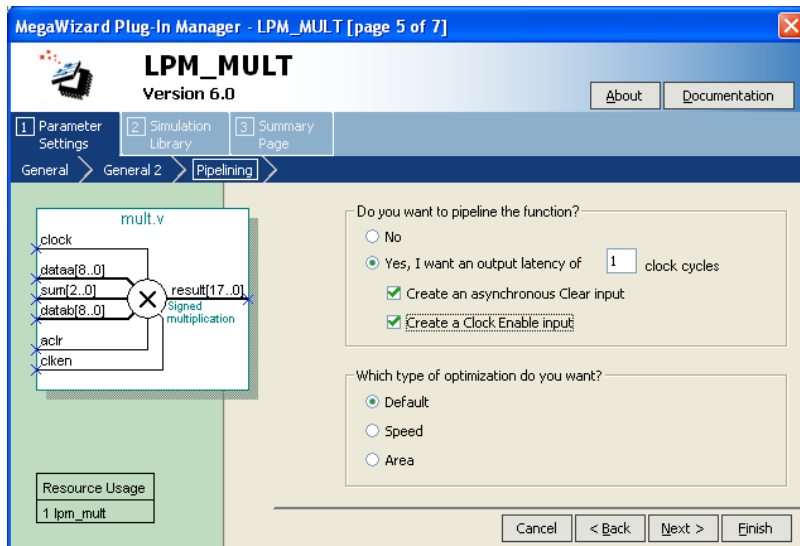


11. On page 3 of the lpm_mult MegaWizard Plug-In Manager, under the Multiplier Configuration section, turn on **Multiply 'dataaa' input by 'datab' input**. Then, to answer the questions **How wide should the 'dataaa' input bus be?** and **How wide should the 'datab' input bus be?**, select 9 bits.
12. Turn on **Create a 'sum' input bus with a width of** and set the width value to 3.
13. To answer the question **How should the width of the 'result' output be determined?**, turn on **Restrict the width to** and set a width value of 18.
14. Click **Next**. Page 4 displays (Figure 2–11).

Figure 2-11. MegaWizard Plug-In Manager—lpm_mult [page 4 of 7]

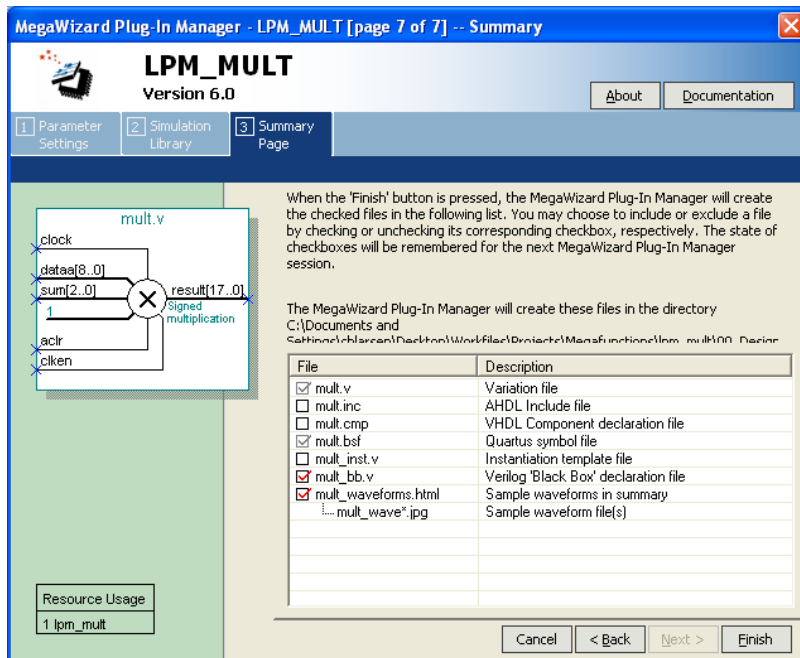


15. On Page 4, to answer **Does the 'dataa' input bus have a constant value?** question, turn on **Yes, the value is** and type 1 in the corresponding field.
16. To answer the question **Which type of multiplication do you want?**, turn on the **Signed** option.
17. To answer the question **Which multiplier implementation should be used?**, turn on the **Use the default implementation** option.
18. Click **Next**. Page 5 displays (Figure 2-12).

Figure 2–12. MegaWizard Plug-In Manager—lpm_mult [page 5 of 7]

19. On Page 5, to answer the question **Do you want to pipeline the function?** turn on **Yes, I want an output latency of** and type 1 in the **Clock cycles** box.
20. Turn on **Create an Asynchronous Clear input** and **Create a Clock Enable input**.
21. To answer **Which type of optimization do you want?**, turn on **Default**.
22. Click **Finish**. Page 7 displays (Figure 2–13).


Figure 2–13. MegaWizard Plug-In Manager—lpm_mult [page 7 of 7] --Summary



23. Page 7 of the lpm_mult MegaWizard Plug-In Manager provides a summary of the megafunction. Ensure that the option to generate the Quartus II software block symbol file (.bsf) is turned on.
24. Click **Finish**. The lpm_mult megafunction is built.
25. Move the pointer to place the multiplier symbol in between the input and output ports of the **multiplier.bdf** file. Click as necessary to place the multiplier symbol.
26. You have now completed the design file (see Figure 2–7 on page 2–11).
27. On the File menu, select **Save** to save the design.


Implement the 9-Bit Multiplier

Next, implement the multiplier, assign the EP2S15F484C3 device to the project, and compile the project.

1. On the Assignments menu, select **Settings** and then **Files**, and add **multiplier.bdf** and **mult.v** to the project. Click **OK**.
2. On the Processing menu, select **Start Compilation**, or click on the compile symbol  to compile the design.
3. In the dialog box asking **Save changes to multiplier.bdf?**, click **Yes**.
4. When the **Full Compilation was successful** message box displays, click **OK**.
5. On the Assignments menu, view how the module is implemented in the Stratix II device by selecting **Timing Closure Floorplan**.

Functional Results—Simulate the 9-Bit Multiplier Design in Quartus

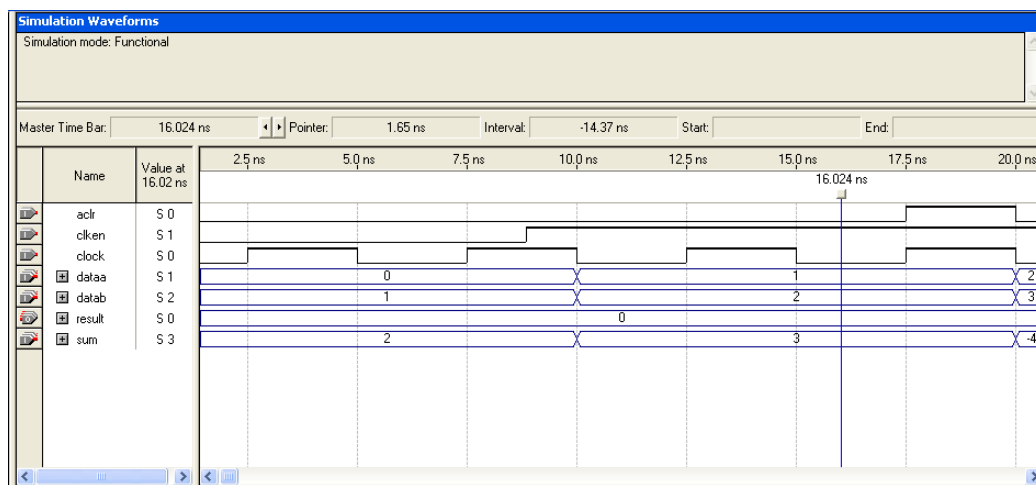
Simulate the `lpm_mult` design module and verify the results. Set up the Quartus II simulator by performing the following steps.

1. On the Processing menu, select **Generate Functional Simulation Netlist** command.
2. When the **Functional Simulation Netlist Generation was successful** message box displays, click **OK**.
3. To open the **Settings** dialog box, on the Assignments menu, select **Settings**.
4. In the Category list, select **Simulator Settings**.
5. Under Simulation mode, select **Functional** and then select the necessary input vector waveform file (**multiplier.vwf**).
6. Click **OK**.
7. On the Processing menu, select **Start Simulation**, or press Ctrl+I, or click on the simulation button  to run a simulation.
8. When the **Simulator was successful** message box displays, click **OK**.

9. In the Simulation Report window, view the simulation output waveforms and verify the results. Figure 2–14 shows the expected simulation results.

These output waveforms show the behavior of `lpm_mult` megafunction for the chosen set of parameters. The design is a 9×9 signed multiplier that produces an 18-bit output. The design has a 3-bit sum input. The output of the multiplier has a latency of 1.

Figure 2–14. Multiplier Simulation Results



Functional Results—Simulate the 9-Bit Multiplier Design in ModelSim-Altera

Simulate the design in ModelSim to compare the results of both simulators.

This User Guide assumes that you are familiar with using ModelSim-Altera before trying out the design example. If you are unfamiliar with ModelSim-Altera, refer to the support page for ModelSim-Altera on the Altera website, www.altera.com. There are various links to topics such as installation, usage, and troubleshooting.

Set up the ModelSim-Altera simulator by performing the following steps.

- a. Unzip the `lpm_mult_msim.zip` file to any working directory on your PC.

10. Browse to the folder in which you unzipped the files and open the **multiplier.do** file in a text editor.
11. In line 1 of the **multiplier.do** file, replace `<insert_directory_path_here>` with the directory path of the appropriate library files. For example, `C:/Modeltech_ae/altera/verilog/stratixii`
12. On the File menu, select **Save**.
13. Start **ModelSim-Altera**.
14. On the File menu, select **Change Directory**.
15. Select the folder in which you unzipped the files. Click **OK**.
16. On the Tools menu, select **Execute Macro**.

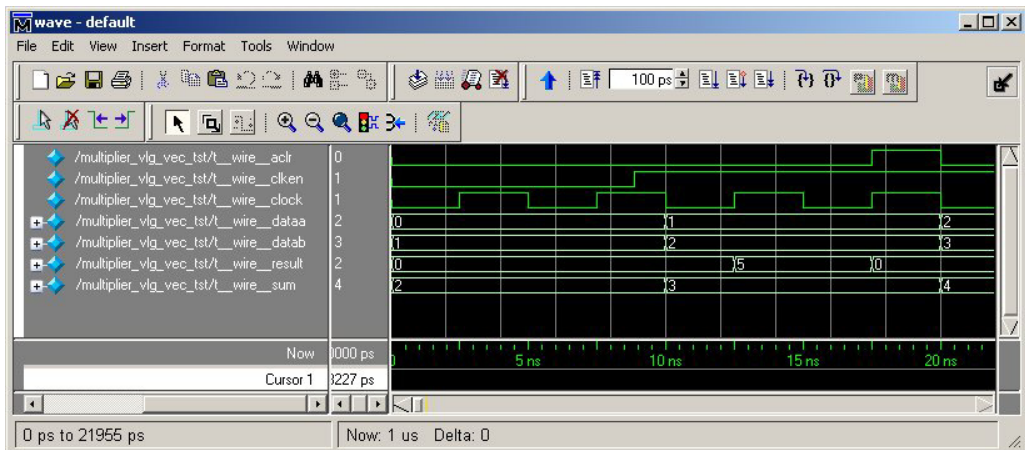
Select the **multiplier.do** file and click **Open**. This is a script file for ModelSim that automates all the necessary settings for the simulation.

17. Verify the results by looking at the **Waveform Viewer** window.

You may need to rearrange signals, remove redundant signals, and change the radix to suit the results in the Quartus II Simulator.

Figure 2–15 shows the expected simulation results in ModelSim.

Figure 2–15. ModelSim Simulation Results



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. You should use these functions during design implementation so you can consistently meet your design goals.

Ports & Parameters

Figure 3–1 shows the ports and parameters for an `lpm_mult` megafunction. Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameterized megafunction interface. The parameter details are only relevant for users who bypass the MegaWizard® Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in their design. The details of these parameters are hidden from MegaWizard Plug-In Manager interface users.



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

Figure 3–1. `lpm_mult` Port & Parameter Description Symbol

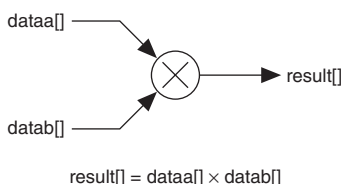


Table 3–1. `lpm_mult` Megafunction Input Ports (Part 1 of 2)

Port Name	Required	Description	Comments
<code>dataa[]</code>	Yes	Value to be multiplied to <code>datab[]</code> : Multiplicand	Input port <code>LPM_WIDTHA</code> wide.
<code>datab[]</code>	Yes	Value to be multiplied to <code>dataa[]</code> : Multiplier	Input port <code>LPM_WIDTHB</code> wide.
<code>sum[]</code>	No	Partial sum. The <code>sum[]</code> input is added to the multiplier output outside of the DSP block.	Input port <code>LPM_WIDTHS</code> wide.
<code>clock</code>	No	A clock port to register and provide pipelined usage.	The clock port provides pipelined operation for the <code>lpm_mult</code> function. For <code>LPM_PIPELINE</code> values other than 0 (default value), the clock port must be connected.

Table 3–1. *lpm_mult* Megafunction Input Ports (Part 2 of 2)

Port Name	Required	Description	Comments
<code>aclr</code>	No	An active high asynchronous clear port for registered usage.	The pipeline initializes to an undefined (X) logic level. The <code>aclr</code> port can be used at any time to reset the pipeline to all 0s, asynchronously to the clock signal.
<code>clken</code>	No	An active high clock enable port for registered usage.	If not used, the default is 1.

Table 3–2. *lpm_mult* Megafunction Output Ports

Port Name	Required	Description	Comments
<code>result[]</code>	Yes	Output port: $\text{result} = \text{dataa}[] \times \text{datab}[] + \text{sum}$. The product LSB is aligned with the <code>sum</code> LSB.	Output port <code>LPM_WIDTHP</code> wide. If <code>LPM_WIDTHP < max(LPM_WIDTHA + LPM_WIDTHB, LPM_WIDTHS)</code> or <code>(LPM_WIDTHA + LPM_WIDTHS)</code> , only the <code>LPM_WIDTHP</code> MSBs are present.

Table 3–3. Parameterized *lpm_mult* Megafunction Interface (Part 1 of 5)

Parameter	Type	Required	Description
<code>LPM_WIDTHA</code>	Integer	Yes	Width of the <code>dataa[]</code> port.
<code>LPM_WIDTHB</code>	Integer	Yes	Width of the <code>datab[]</code> port.
<code>LPM_WIDTHP</code>	Integer	Yes	Width of the <code>result[]</code> port.
<code>LPM_WIDTHS</code>	Integer	Yes	Width of the <code>sum[]</code> port. Required even if the <code>sum[]</code> port is not used.
<code>LPM_REPRESENTATION</code>	String	No	Data representation: "SIGNED", "UNSIGNED", or "UNUSED". The default is "UNSIGNED". The signed representation for all library of parameterized modules (LPM) megafunctions is two's complement.

Table 3–3. Parameterized *lpm_mult* Megafunction Interface (Part 2 of 5)

Parameter	Type	Required	Description
LPM_PIPELINE	Integer	No	Specifies the number of clock cycles of latency associated with the <code>result[]</code> output. A value of zero (0) indicates that no latency exists, and that a purely combinational function is instantiated. If the value of the LPM_PIPELINE parameter is greater than zero for the Mercury™ dedicated multiplier, one of the pipeline stages is always placed on the outputs. The default is 0 (non-pipelined). For HardCopy® Stratix®, Stratix, and Stratix GX devices, if the design uses DSP blocks, you can increase the performance of the design when the value of the LPM_PIPELINE parameter is three or less.
LPM_HINT	String	No	Assigns Altera®-specific parameters in VHDL Design Files (.vhd). The default is "UNUSED".
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.
INPUT_A_IS_CONSTANT	String	No	Values are "YES", "NO", and "UNUSED". If <code>dataa[]</code> is connected to a constant value, setting INPUT_A_IS_CONSTANT to "YES" optimizes the multiplier for resource usage and speed. The default is "NO".
INPUT_B_IS_CONSTANT	String	No	Values are "YES", "NO", and "UNUSED". If <code>datab[]</code> is connected to a constant value, setting INPUT_B_IS_CONSTANT to "YES" optimizes the multiplier for resource usage and speed. The default is "NO".

Table 3–3. Parameterized *lpm_mult* Megafunction Interface (Part 3 of 5)

Parameter	Type	Required	Description
USE_EAB	String	No	Values are "ON", "OFF", and "UNUSED". Setting the USE_EAB parameter to "ON" lets the Quartus II software use ESBs to implement 4×4 or $(8 \times \text{const value})$ building blocks in APEX™ 20K, APEX II, Excalibur™, and Mercury devices, or EABs in ACEX® 1K and FLEX 10K® devices. Altera recommends that you set USE_EAB to "ON" only when LCELLS are in short supply. This parameter is not available for simulation with other EDA simulators. If you wish to use this parameter when you instantiate the function in a Block Design File (.bdf), you must specify it by entering the parameter name and value manually with the Parameters tab (Symbol Properties Command) or the Parameters tab (Block Properties Command). You can also use this parameter name in a Text Design File (.tdf) or a Verilog Design File (.v). You must use the LPM_HINT parameter to specify the USE_EAB parameter in VHDL Design Files.
LATENCY	Integer	No	Same as LPM_PIPELINE. This parameter is provided only for backward compatibility. For all new designs, you must use the LPM_PIPELINE parameter instead.

Table 3–3. Parameterized lpm_mult Megafunction Interface (Part 4 of 5)

Parameter	Type	Required	Description
MAXIMIZE_SPEED	Integer	No	You can specify a value between 0 and 10. If used, the Quartus II software attempts to optimize a specific instance of the lpm_mult function for speed rather than area, and overrides the setting of the Optimization Technique logic option. If MAXIMIZE_SPEED is unused, the value of the Optimization Technique option is used instead. For a "SIGNED" multiplier with no inputs being a constant, if the setting for MAXIMIZE_SPEED is 9–10, the compiler optimizes the lpm_mult megafunction for larger area. These settings are for backward compatibility only; if the setting is between 6 and 8, the compiler optimizes for larger area and higher speed; if the setting is between 1 and 5, the compiler optimizes for smaller area and high speed. If the setting is 0, the smallest and, generally, slowest design results. For designs with LPM_WIDTHHB parameters that are non-power-of-2, the default setting is 1 through 5. For designs with LPM_WIDTHHB parameters that are a power-of-2, the default is 6–8. For an "UNSIGNED" multiplier with no inputs being a constant, if the setting for MAXIMIZE_SPEED is 6 or higher, the compiler optimizes for larger area and higher speed. If the setting is between 0 and 5, which is the default value, the compiler optimizes for smaller area.
DEDICATED_MULTIPLIER_CIRCUITRY	String	No	Specifies whether to use dedicated multiplier circuitry. Values are "AUTO", "YES", and "NO". The default is "AUTO". This parameter is available for Mercury, HardCopy Stratix, Stratix, Stratix II, Stratix GX, and Cyclone™ II devices only. For HardCopy Stratix, Stratix, and Stratix GX devices, the value of "AUTO" specifies that the Quartus II software makes a choice whether to use the dedicated multiplier circuitry based on the width of the multiplier. For Mercury devices, a value of "AUTO" defaults to no dedicated multiplier circuitry.

Table 3–3. Parameterized lpm_mult Megafunction Interface (Part 5 of 5)

Parameter	Type	Required	Description
DEDICATED_MULTIPLIER_MIN_INPUT_WIDTH_FOR_AUTO	Integer	No	If the DEDICATED_MULTIPLIER_CIRCUITRY parameter setting is "AUTO", this parameter specifies the minimum value of the LPM_WIDTHA and LPM_WIDTHB parameters for the multiplier to be built using dedicated circuitry. This parameter is available for HardCopy Stratix, Mercury, Stratix, Stratix II, Stratix GX, and Cyclone II devices only.
DEDICATED_MULTIPLIER_MIN_OUTPUT_WIDTH_FOR_AUTO	Integer	No	If the DEDICATED_MULTIPLIER_CIRCUITRY parameter setting is "AUTO", this parameter specifies the minimum value of the sum of the LPM_WIDTHA and LPM_WIDTHB parameters for the multiplier to be built using dedicated circuitry. This parameter is available for HardCopy Stratix, Mercury, Stratix, Stratix II, Stratix GX, and Cyclone II devices only.