lpm_add_sub Megafunction

User Guide

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

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

UG-MF9804-2.2

Contents

About this User Guide

Chapter 1. About this Megafunction

Chapter 2. Getting Started

Chapter 3. Specifications

About this User Guide

Revision History

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

Date/Version	Changes Made	Summary of Changes
March 2007 v2.2	● Added Cyclone® III device to list of supported devices.	Updated for Quartus® II version 7.0 by adding support for Cyclone III device.
December 2006 v2.1	● Minor updates to add Stratix® III device information	
May 2006 v2.0	● Revised for Quartus II® 6.0 software release	
September 2004 v1.0	● 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 and Canada	All Other Locations
Technical support	www.altera.com/mysupport/	altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	(408) 544-7000 *(1)* (7:00 a.m. to 5:00 p.m. Pacific Time)
Product literature	www.altera.com	www.altera.com
Altera literature services	lit_req@altera.com *(1)*	lit_req@altera.com *(1)*
Non-technical customer service	(800) 767-3753 (7:00 a.m. to 5:00 p.m. Pacific Time)	(408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time)
FTP site	ftp.altera.com	ftp.altera.com

Note to table:

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

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: **f_{MAX}**, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>***.pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input`. Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user’s work.
⚠ WARNING	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 on a particular topic.

Chapter 1. About this Megafunction

Device Family Support

The lpm_add_sub megafunction supports the following target Altera device families:

- Stratix® III
- Stratix II
- Stratix II GX
- Stratix
- Stratix GX
- Cyclone® III
- Cyclone II
- Cyclone
- HardCopy® II
- HardCopy Stratix
- MAX® II
- MAX 7000AE
- MAX 7000B
- MAX 7000S
- MAX 3000A
- 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 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 simply setting parameters.

Features

The lpm_add_sub megafunction implements adder and subtractor functions and offers many additional features, which include:

- Selecting input data widths
- Implementing addition only, subtraction only, or variable using the `add_sub` input port
- Specifying constant values on either one of the input buses
- Selecting carry/borrow-out input
- Selecting carry/borrow-in output
- Selecting overflow output
- Pipelining with output latency
- Active low asynchronous clear and clock enable inputs

General Description

The lpm_add_sub megafunction is one of the arithmetic megafunctions provided in the Quartus® II MegaWizard® Plug-In Manager. The lpm_add_sub megafunction lets you implement an adder or a subtractor to add or subtract sets of data to produce an output containing the sum or difference of both values. Figure 1–1 shows a logic diagram of an adder/subtractor with pipelining.

Figure 1–1. Block Diagram of an Adder/Subtractor Function

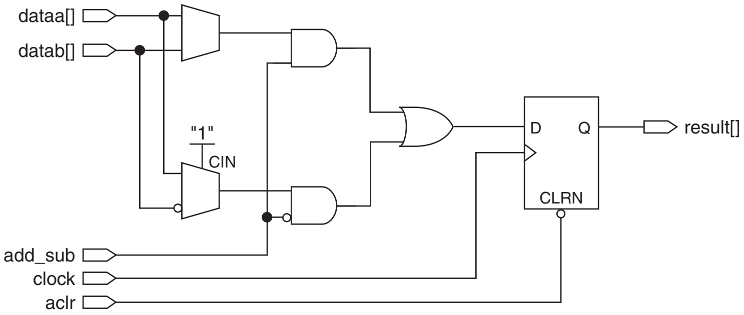
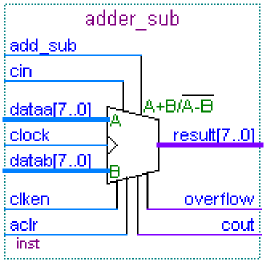


Figure 1–2 shows the lpm_add_sub megafunction symbol.

Figure 1–2. lpm_add_sub Megafunction Symbol



Common Applications

Designing with the lpm_add_sub megafunction is an efficient method of implementing an adder or subtractor in Altera devices. An adder/subtractor performs basic mathematical functions often found in digital signal processing (DSP) applications.

Resource Utilization and Performance

The lpm_add_sub megafunction is implemented in adaptive look-up tables (ALUTs) in Stratix II devices and in logic elements (LEs) in the Stratix, Stratix GX, HardCopy Stratix, Cyclone II, and Cyclone devices.

Table 1–1 summarizes the resource usage for an lpm_add_sub function used to implement an 8-bit unsigned adder.

Table 1–1. lpm_add_sub Megafunction Resource Usage

Device Family	Optimization	Width	Logic Usage
Stratix II	Speed, Balanced, and Area	8 bits	8 ALUTS
Stratix, Stratix GX, Cyclone II, Cyclone, and Hardcopy Stratix	Speed, Balanced, and Area	8 bits	8 logic elements (LEs)

Chapter 2. Getting Started

Software and System Requirements

The instructions in this section require the following software:

- For operating system support information, refer to:

http://www.altera.com/support/software/os_support/oss-index.html

- Quartus® II software beginning with version 6.1

MegaWizard Plug-In Manager Customization

You can use the MegaWizard® Plug-In Manager to specify the lpm_add_sub megafunction features for each adder/subtractor in the design.

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

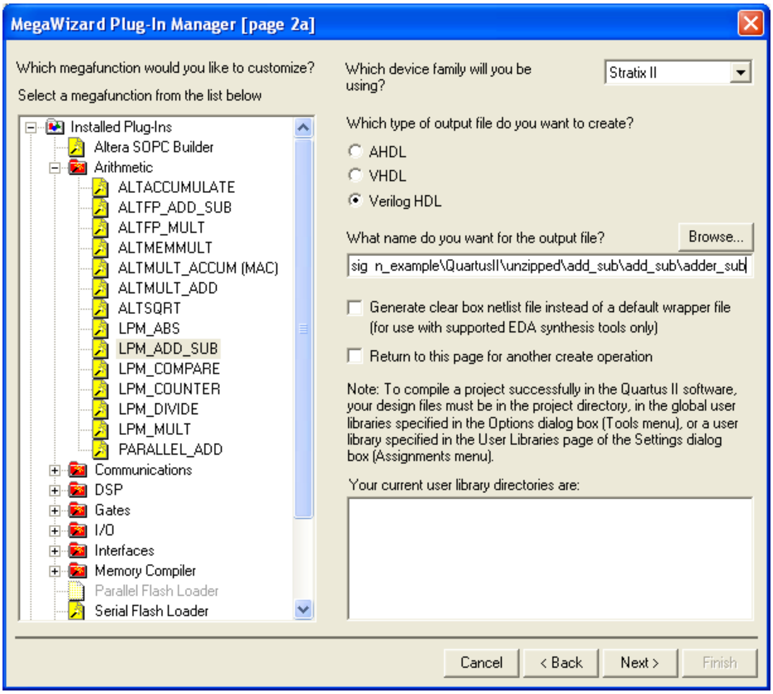
- On the Tools menu, choose the **MegaWizard Plug-In Manager** command.
- When working in the Block Editor, click **MegaWizard Plug-In Manager** in the **Symbol** window.
- Start the stand-alone version of the **MegaWizard Plug-In Manager** by typing the following command at the command prompt: `qmegawiz`↵

Using the MegaWizard Plug-In Manager

This section provides descriptions for the options available in the lpm_add_sub megafunction wizard.

On Page 2a, select the lpm_add_sub 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_ADD_SUB [Page 2a]



Which megafunction would you like to customize?

Select a megafunction from the list below

Which device family will you be using? Stratix II

Which type of output file do you want to create?

○ AHDL

○ VHDL

● Verilog HDL

What name do you want for the output file?

sig n_example\QuartusII\unzipped\add_sub\add_sub\adder_sub

□ Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only)

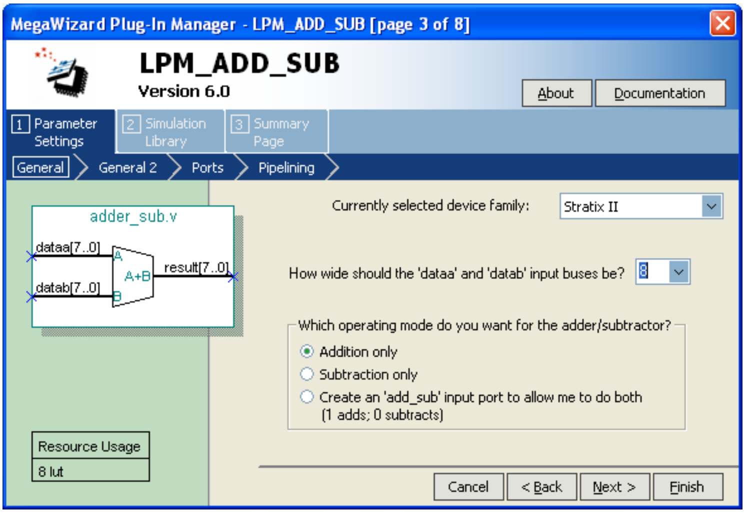
□ Return to this page for another create operation

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

Your current user library directories are:

On Page 3 of the lpm_add_sub wizard, specify an add operation, a subtract operation, or both operations together, and set the input data widths. Figure 2–2 shows Page 3 of the lpm_add_sub megafunction wizard.

Figure 2–2. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 3 of 8]



Currently selected device family: Stratix II

How wide should the 'dataa' and 'datab' input buses be? 8

Which operating mode do you want for the adder/subtractor?

● Addition only
○ Subtraction only
○ Create an 'add_sub' input port to allow me to do both (1 adds; 0 subtracts)

Table 2–1 shows the features and settings of the lpm_add_sub megafunction. Use this table, along with the hardware descriptions for the features, to determine the appropriate settings.

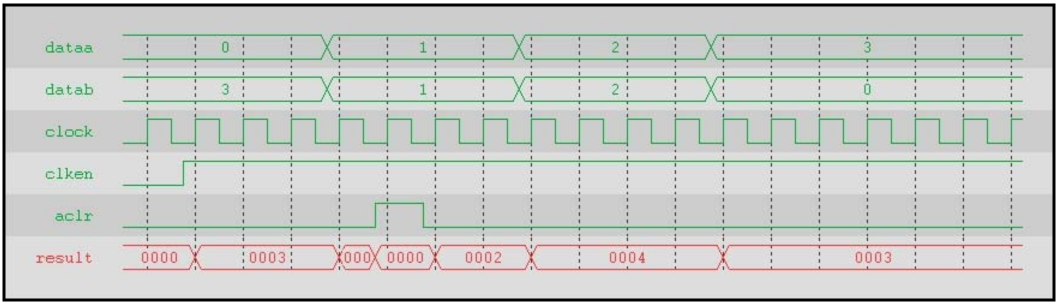
Table 2–1. lpm_add_sub MegaWizard Plug-in Manager Page 3 Options

Function	Description
How wide should the 'dataa' and 'datab' input buses be?	Specify the width of the data to be added or subtracted.
Which operation mode do you want for the adder/subtractor?	Specify which operation you want to perform on the data.

To generate a sample simulation waveform or launch the Quartus II Help, from Page 3 of the megafunction wizard, click the **Documentation** button and select **Generate Sample Waveforms** or **Quartus II Megafunction Reference** (see Figure 2–2).

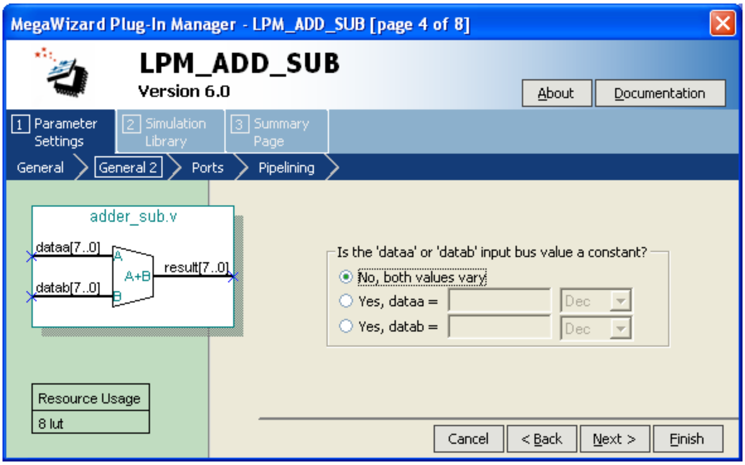
The sample waveform shown in Figure 2–3 illustrates the behavior of the lpm_add_sub megafunction for the chosen set of parameters in the lpm_add_sub design module. This option generates a sample waveform in an HTML- format file in the specified design directory. The HTML file contains descriptions showing the adder/subtractor operation.

Figure 2–3. Sample Waveforms for the lpm_add_sub Megafunction



On Page 4 of the lpm_add_sub megafunction wizard, specify whether one or both of the input buses will be a constant; and if so, with what value (Figure 2–4).

Figure 2–4. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 4 of 8]

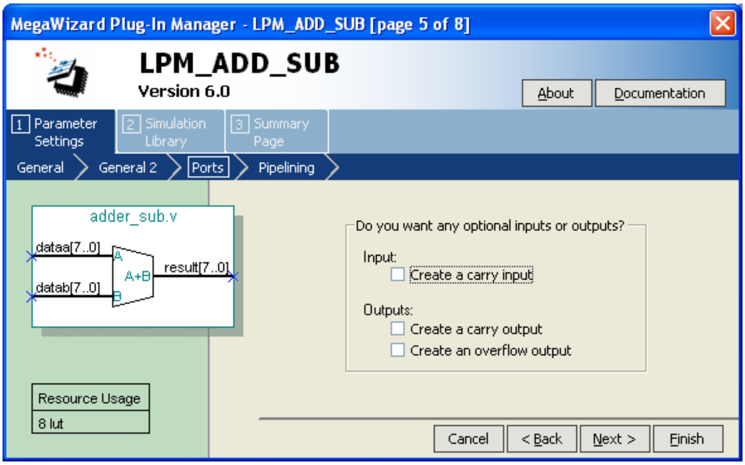


Is the 'dataa' or 'datab' input bus value a constant?

● No, both values vary
○ Yes, dataa = Dec
○ Yes, datab = Dec

On Page 5 of the lpm_add_sub megafunction wizard, specify whether to implement a carry input, a carry output, or an overflow output (Figure 2–5).

Figure 2–5. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 5 of 8]



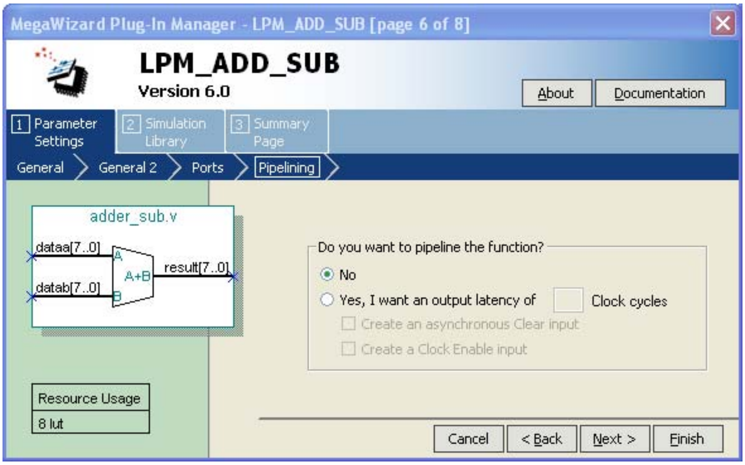
Do you want any optional inputs or outputs?

Input:
□ Create a carry input

Outputs:
□ Create a carry output
□ Create an overflow output

On Page 6 of the lpm_add_sub megafunction wizard, specify whether to pipeline the function, and if so, whether to create an asynchronous Clear input and a Clock Enable input for the registers in the function. You can also specify the amount of output latency in clock cycles (Figure 2–6).

Figure 2–6. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 6 of 8]



Do you want to pipeline the function?

● No

○ Yes, I want an output latency of ☐ Clock cycles

☐ Create an asynchronous Clear input

☐ Create a Clock Enable input

Resource Usage
8 lut

Inferring Megafunctions from HDL Code

Synthesis tools, including the 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/or 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

The Quartus II software performs analysis and elaboration to build the structure of your design during compilation. 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), 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

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 delays 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 uses 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, you must use the synthesis attributes `keep` or `preserve`. These are Verilog and VHDL synthesis attributes that direct analysis and synthesis to keep a particular wire, register, or node intact. You can 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 volume 1 of the *Quartus II Handbook*.

EDA Simulation

To simulate designs containing LPMs or MegaWizard-generated functions, use the Altera functional simulation models listed below:

- **220model.v** (for Verilog HDL)
- **220pack.vhd** and **220model.vhd** (for VHDL)

If you are simulating a design that uses VHDL-1987 with the Quartus II software, use **220model_87.vhd**. These files include simulation models for standard LPM functions. For more information about LPMs, see the Quartus II Help. The model files in the Quartus II software are located in the <***Quartus II installation directory***>**\eda\sim_lib\** directory.

For more information about Simulation, refer to volume 3 of the *Quartus II Handbook*. This volume shows you 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 those files are located.

SignalTap II Embedded Logic Analyzer

The SignalTap® II embedded logic analyzer provides 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, you must 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 with your design in the selected device seamlessly.

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

Design Example: 8-Bit Multiplier-Adder

This section presents a design example that uses the lpm_add_sub megafunction to generate a basic multiplier-adder. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you proceed through the wizard, each page is described in detail. When you are finished with this example, you can incorporate it into your overall project.

Design Files

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

Example

The objective of this design is to instantiate an lpm_add_sub megafunction using the MegaWizard Plug-In Manager. In this case, an adder is instantiated after a multiplier to create a multiplier adder that is commonly used in DSP functions. You can change the parameters as needed for your design. In this example, you will perform the following activities:

- Generate an 8-bit adder design module using the lpm_add_sub megafunction in the MegaWizard Plug-In Manager
- Implement the multiplier adder design in an Altera device by assigning the Stratix II EP2S15F484C3 device and compiling the project
- Simulate the customized multiplier adder design

Generate a 16-Bit Adder

1. Open the project file **mult_adder.qpf**.

2. Open the top-level file **mult_adder.bdf**. This is an incomplete file. You will be adding the adder to the multiplier to create the multiplier-adder.

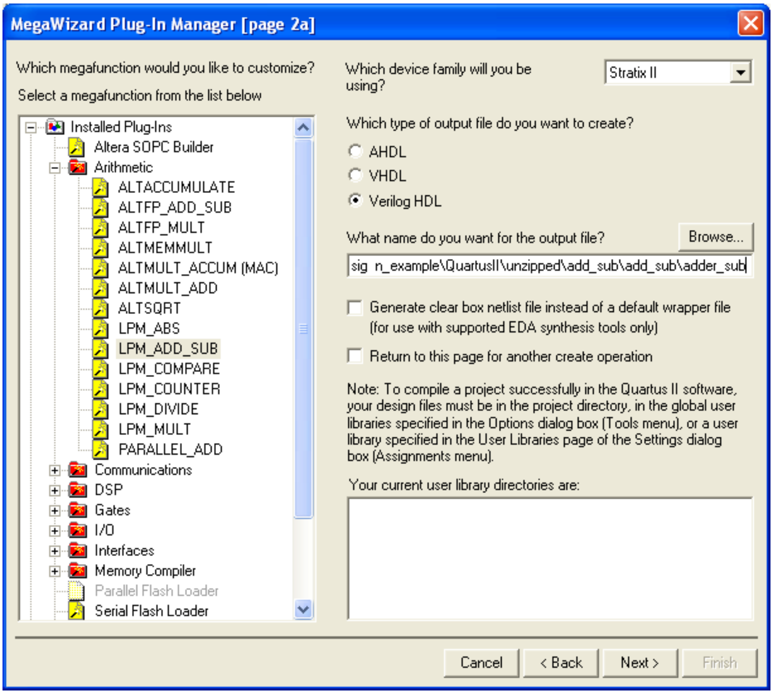
3. Double-click on a blank area in the block design (**.bdf**) file.

4. Click **MegaWizard Plug-In Manager** in the **Symbol** window.

5. In the **What action do you want to perform?** section on Page 1 of the megafunction wizard, select **Create a new custom megafunction variation**.

6. Click **Next**. Page 2a appears (Figure 2–7).

Figure 2–7. MegaWizard Plug In Manager [Page 2a]



MegaWizard Plug-In Manager [page 2a]

Which megafunction would you like to customize?

Select a megafunction from the list below

Which device family will you be using? Stratix II

Which type of output file do you want to create?

○ AHDL
○ VHDL
● Verilog HDL

What name do you want for the output file? sig n_example\QuartusII\unzipped\add_sub\add_sub\adder_sub

□ Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only)

□ Return to this page for another create operation

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

Your current user library directories are:

7. On Page 2a, expand the Arithmetic folder and select the `LPM_ADD_SUB` megafunction.

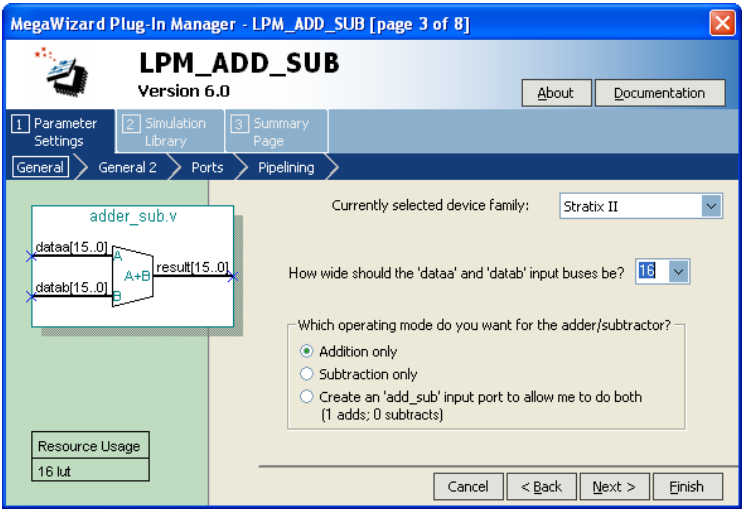
8. In the **Which device family will you be using?** list, select **Stratix II**.

9. Under the **Which type of output file do you want to create?** section, select **Verilog HDL**.

10. Set the output file name to **<*project directory*>\adder_sub**.

11. Click **Next**. Page 3 appears (Figure 2–8).

Figure 2–8. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 3 of 8]



MegaWizard Plug-In Manager - LPM_ADD_SUB [page 3 of 8]

LPM_ADD_SUB
Version 6.0

Currently selected device family: Stratix II

How wide should the 'dataa' and 'datab' input buses be? 16

Which operating mode do you want for the adder/subtractor?
- ● Addition only
- ○ Subtraction only
- ○ Create an 'add_sub' input port to allow me to do both (1 adds; 0 subtracts)

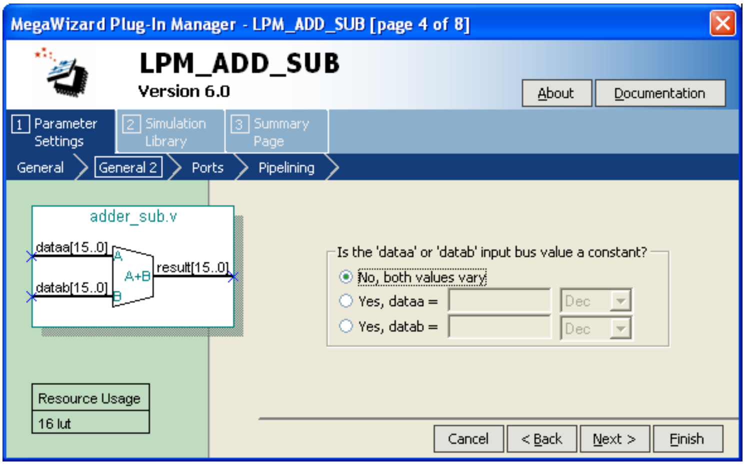
Resource Usage
16 lut

12. On Page 3 of the lpm_add_sub megafunction wizard, in the **How wide should the 'dataa' and 'datab' input buses be?** section, select **16** bits (Figure 2–8).

13. In the **Which operating mode do you want for the adder/subtractor?** section, select **Addition only**.

14. Click **Next**. Page 4 appears (Figure 2–9).

Figure 2–9. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 4 of 8]



Is the 'dataa' or 'datab' input bus value a constant?

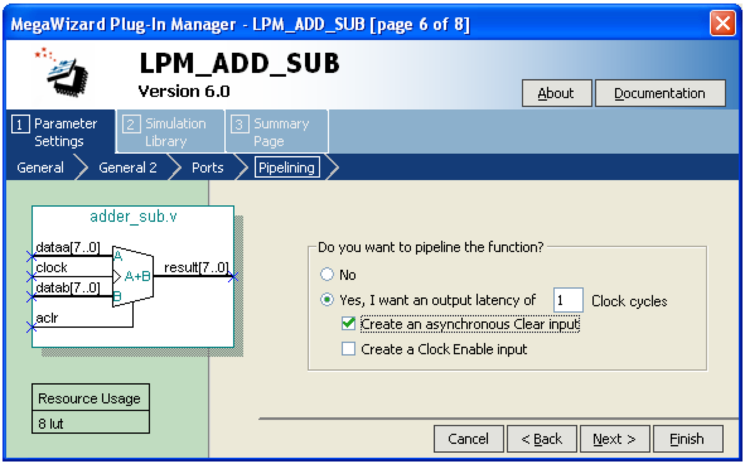
● No, both values vary
○ Yes, dataa = Dec
○ Yes, datab = Dec

15. On Page 4, under **Is the 'dataa' or 'datab' input bus value a constant?**, make sure **No, both values vary** is selected.

16. Click **Next** twice to skip Page 5 and go to Page 6.

17. Under **Do you want to pipeline the function**, select **Yes, I want an output latency of 1 Clock cycles** (make sure **1** is selected). Also select **Create an asynchronous Clear input** (Figure 2–10).

Figure 2–10. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 6 of 8]



Do you want to pipeline the function?

○ No

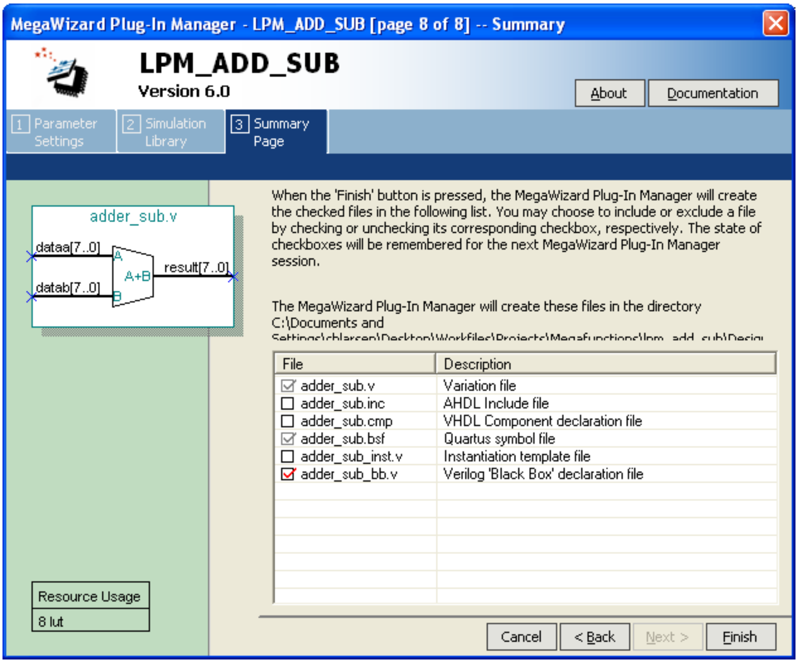
● Yes, I want an output latency of 1 Clock cycles

☑ Create an asynchronous Clear input

☐ Create a Clock Enable input

18. Click **Finish**, and leave all the options in their default state. This example does not use Pages 5 or 7. Page 8 appears (Figure 2–11).

Figure 2–11. MegaWizard Plug-In Manager LPM_ADD_SUB [Page 8 of 8]



When the 'Finish' button is pressed, the MegaWizard Plug-In Manager will create the checked files in the following list. You may choose to include or exclude a file by checking or unchecking its corresponding checkbox, respectively. The state of checkboxes will be remembered for the next MegaWizard Plug-In Manager session.

The MegaWizard Plug-In Manager will create these files in the directory C:\Documents and Settings\chlarsen\Desktop\Workfiles\Projects\Megafunctions\lpm_add_sub\Desig

File	Description
☑ adder_sub.v	Variation file
☐ adder_sub.inc	AHDL Include file
☐ adder_sub.cmp	VHDL Component declaration file
☑ adder_sub.bsf	Quartus symbol file
☐ adder_sub_inst.v	Instantiation template file
☑ adder_sub_bb.v	Verilog 'Black Box' declaration file

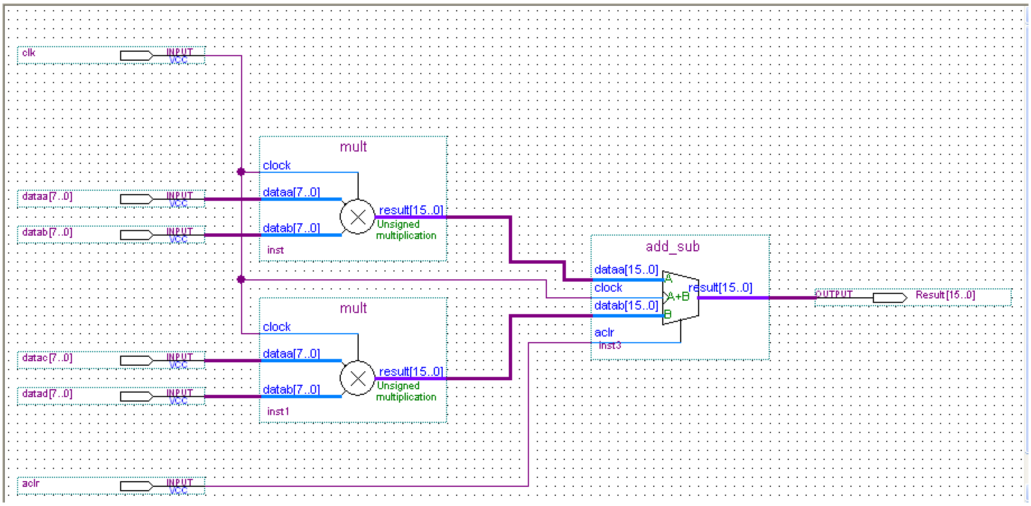
19. On Page 8 of the megafunction wizard, ensure that the option to generate the Quartus II software block symbol file (**.bsf**) is selected (Figure 2–11).

20. Click **Finish**. The lpm_add_sub megafunction is built.

21. Move the pointer to place the adder symbol in between the input and output ports of the **mult_adder.bdf** file. Click to place the adder symbol.

    You have now completed the design file (Figure 2–12).

Figure 2–12. Completed mult_adder Block Diagram



19. On the File menu, select **Save** to save the design.

Implement the 8-Bit Multiplier-adder

Assign the EP2S15F484C3 device to the project and compile the project.

1. On the Assignments menu, select **Settings** to open the **Settings** dialog box.

2. Click the **Devices** category. In the Family field, ensure that Stratix II is selected.

3. In the T**arget device** section, from the **Available devices** list, select **EP2S15F484C3**.

4. Click **OK**.

5. To compile the design, on the Processing menu, select **Start Compilation**.

6. When the **Full Compilation was successful** message box appears, click **OK**.

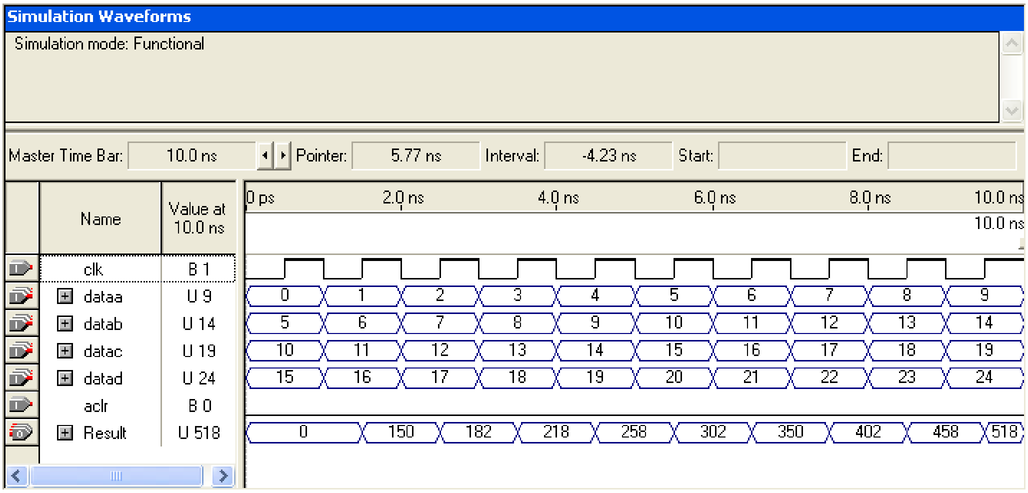
7. To view how the multiplier-adder is implemented in the Stratix II device, on the Assignments menu, select **Timing Closure Floorplan**.

Functional Results—Simulate the 8-Bit Multiplier-adder Design in Quartus

Simulate the design to verify the results. Setup the Quartus II Simulator by performing the following steps.

1. On the Processing menu, select **Generate Functional Simulation Netlist**.
2. When the generation is finished, you will see a **Functional Simulation Netlist Generation was successful** message box**.** Click **OK**.
3. On the **Assignments** menu, click **Settings**. The **Settings** dialog box appears.
4. In the **Category** list, select **Simulator Settings**.
5. In the **Simulation mode** list, select **Functional**.
6. In the Simulation mode section, set the path to the ***<project directory>*****mult_adder.vwf** file.
7. Click **Open**.
8. In the **How do you want to determine the end time for the simulation?** section, select **Run simulation until all vector stimuli have been used**.
9. Click **OK**.
10. On the Processing menu, select **Start Simulation** to start the simulation.
11. When the **Simulator was successful** message box appears, click **OK**.
12. In the **Simulation Report** window, look at the simulation output waveform and verify the results. Figure 2–13 shows the expected simulation results of multiplier adder function.

Figure 2–13. Simulation Waveforms



Simulation mode: Functional

Master Time Bar: 10.0 ns Pointer: 5.77 ns Interval: -4.23 ns Start: End:

Name	Value at 10.0 ns	Waveform
clk	B 1	
dataa	U 9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
datab	U 14	5, 6, 7, 8, 9, 10, 11, 12, 13, 14
datac	U 19	10, 11, 12, 13, 14, 15, 16, 17, 18, 19
datad	U 24	15, 16, 17, 18, 19, 20, 21, 22, 23, 24
aclr	B 0	
Result	U 518	0, 150, 182, 218, 258, 302, 350, 402, 458, 518

Functional Results—Simulate the 8-Bit Multiplier-Adder 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, refer to http://www.altera.com/support/ software/products/modelsim/mod-modelsim.html, which is a support page for ModelSim-Altera. There are various links to topics such as installation, usage, and troubleshooting.

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

1. Unzip the **mult_adder_msim.zip** file to any working directory on your PC.

2. Browse to the folder in which you unzipped the files and open the **mult_adder.do** file in a text editor.

3. In line 1 of the **mult_adder.do** file, replace <*insert_directory_path_here*> with the directory path of the appropriate library files. For example, `C:/Modeltech_ae/altera/verilog/stratixii`

4. On the File menu, select **Save**.

5. Start **ModelSim-Altera**.

6. On the File menu, select **Change Directory**.

7. Select the folder in which you unzipped the files. Click **OK**.

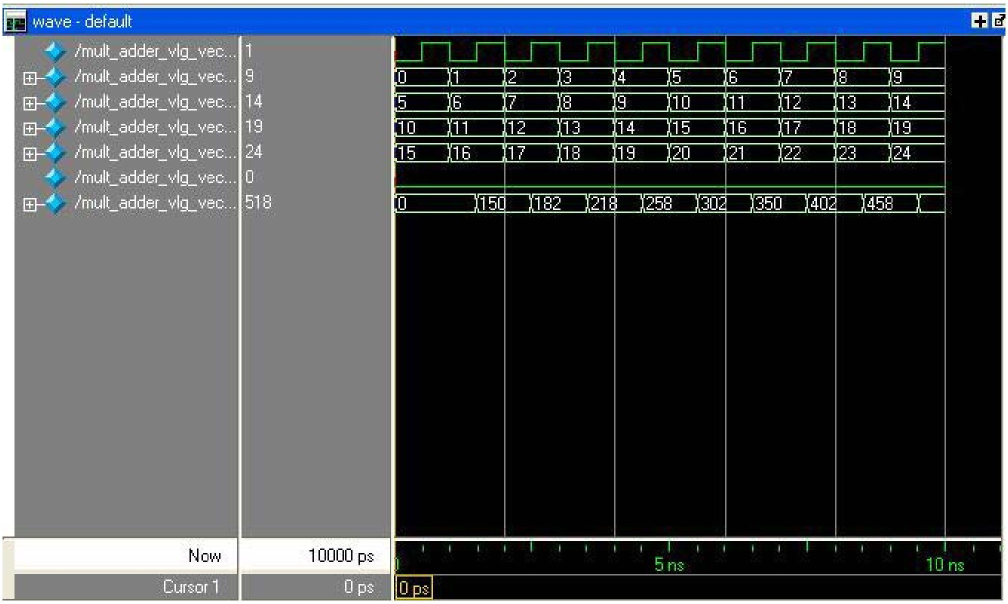
8. On the Tools menu, select **Execute Macro**.

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

10. 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 2–14 shows the expected simulation results in ModelSim.

Figure 2–14. 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.

Chapter 3. Specifications

Ports and Parameters

Figure 3–1 below shows the ports and parameters for the lpm_add_sub megafunction. Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the lpm_add_sub megafunction parameters.

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

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.

Figure 3–1. lpm_add_sub Port and Parameter Description Symbol

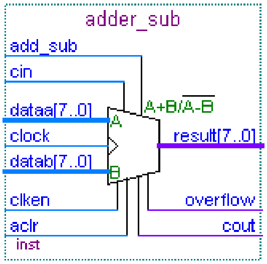


Table 3–1. lpm_add_sub Megafunction Input Ports

Name	Required	Description	Comment
cin	No	Carry-in to the low-order bit.	If omitted, the default is 0.
dataa[]	Yes	Augend/minuend.	Input port LPM_WIDTH wide.
add_sub	No	If the signal is high, the operation = dataa[]+datab[]+cin. If the signal is low, the operation = dataa[]-datab[]+cin-1.	If you use the LPM_DIRECTION parameter, lpm_add_sub cannot be used. If omitted, the default is ADD. You should use the LPM_DIRECTION parameter to specify the operation of the lpm_add_sub function, rather than assigning a constant to the add_sub port.
clock	No	Clock enable for pipelined use.	You can use the aclr port at any time to reset the pipeline to all 0s, asynchronously to the clock signal. Active high clear.
aclr	No	Asynchronous clear for pipelined use.	The pipeline initializes to an undefined (X) logic level. The aclr port can be used at any time to reset the pipeline to all 0s, asynchronously to the clock signal.

Table 3–2. lpm_add_sub Megafunction Output Ports

Name	Required	Description	Comment
result[]	Yes	dataa[]+datab[]+cin or dataa[]-datab[]+cin-1.	Output port LPM_WIDTH wide.
cout	No	Carry-out (borrow-in) of the most significant bit MSB.	The cout port has a physical interpretation as the carry-out (borrow-in) of the MSB. The cout port is most meaningful for detecting overflow in UNSIGNED operations. The cout port operates in the same manner for SIGNED and UNSIGNED operations.
overflow	No	Result exceeds available precision.	The overflow port has a physical interpretation as the XOR of the carry-in to the MSB while the carry-out of the MSB. The overflow port is meaningful only when the LPM_REPRESENTATION parameter value is SIGNED.

Table 3–3. lpm_add_sub Megafunction Parameters

Name	Type	Required	Comment
`LPM_WIDTH`	Integer	Yes	Specifies the width of the `dataa[]`, `datab[]`, and `result[]` ports.
`LPM_DIRECTION`	String	No	Values are `ADD`, `SUB`, and `UNUSED`. If omitted, the default is `DEFAULT`, which directs the parameter to take its value from the `add_sub` port. The `add_sub` port cannot be used if `LPM_DIRECTION` is used. You should use the `LPM_DIRECTION` parameter to specify the operation of the lpm_add_sub function, rather than assigning a constant to the `add_sub` port.
`LPM_REPRESENTATION`	String	No	Type of addition performed: `SIGNED`, `UNSIGNED`, or `UNUSED`. If omitted, the default is `SIGNED`. The signed representation for all library of parameterized modules (LPM) megafunctions is two's complement.
`LPM_PIPELINE`	Integer	No	Specifies the number of Clock cycles of latency associated with the `result[]` output. A value of zero (0) indicates that no latency exists, and that a purely combinational function will be instantiated. If omitted, the default is 0 (non-pipelined).