



altaccumulate Megafunction

User Guide



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

Software Version: 7.0
Document Version: 2.3
Document Date: March 2007

Copyright © 2007 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-MF83104-2.3



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	1-2
Features	1-2
General Description	1-2
Common Applications	1-3
Resource Utilization & 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	2-5
Instantiating Megafunctions in HDL Code	2-5
Identifying a Megafunction after Compilation	2-5
Simulation	2-5
Quartus II Simulation	2-5
EDA Simulation	2-6
SignalTap II Embedded Logic Analyzer	2-6
Design Example: 8-Bit Multiplier Accumulator	2-7
Design Files	2-7
Example	2-7
Generate an 8-Bit Accumulator	2-8
Implement the 8-Bit Multiplier-Accumulator	2-13
Simulate the 8-Bit Multiplier-Accumulator Design in Quartus II	2-13
Simulate the 8-Bit Multiplier-Accumulator Design in ModelSim-Altera	2-15
Conclusion	2-17
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	Version	Changes Made
March 2007	2.3	Minor update to add Cyclone III device information; no new screenshots were added
December 2006	2.2	Minor update to add Stratix III device information; no new screenshots were added
May 2006	2.1	Minor update
April 2006	2.0	Updated for Quartus® 6.0, including: <ul style="list-style-type: none">• Steps and screenshots updated for Quartus II 6.0 version software• New section on ModelSim simulation
January 2006	1.1	Updated for Quartus® 5.1, including: <ul style="list-style-type: none">• Steps and screenshots updated for Quartus II 5.1 version software• Removed Timing Wizard
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	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	+1 408-544-8767 (1) 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com (1)	www.altera.com (1)
Altera literature services	lliterature@altera.com	literature@altera.com
Non-technical customer service	(800) 767-3753	+1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) 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} , 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 on a particular topic.



Chapter 1. About This Megafunction

Device Family Support

The altaccum megafunctions support 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 megafunctions with adjustable parameters that optimizes with 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 altaccumulate megafunction implements an accumulator and offers many additional features, which include:

- Parameterizable input data widths
- Support for pipelining with parameterized output latency
- Active high asynchronous clear and clock enable control inputs

General Description

The altaccumulate megafunction is an arithmetic megafunction provided in the Quartus® II software MegaWizard® Plug-In Manager.

Accumulators are common basic building blocks useful in digital signal processing (DSP) functions. The altaccumulate megafunction implements an accumulator that adds its input data to the registered result of its previous operation, allowing the accumulator to store the sum of a serial data stream. Figure 1-1 illustrates a basic, 8-bit, addition-based accumulator.

Figure 1-1. 8-Bit Accumulator Block Diagram

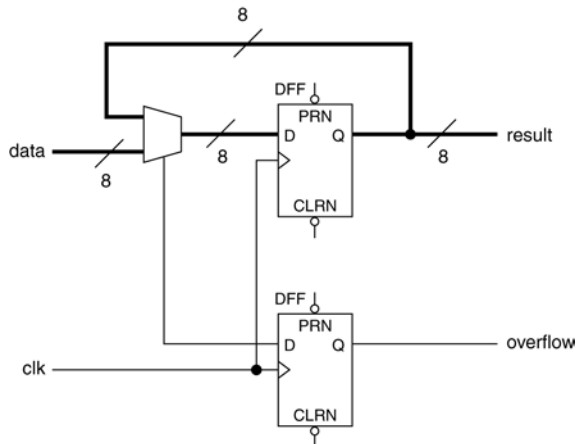
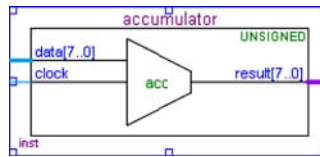


Figure 1–2. Basic altaccumulate Symbol



The altaccumulate megafunction implements a basic accumulator and offers additional features, including:

- Parameterizable input and output data widths
- Parameterizable output types (signed, unsigned, or the ability to create a `sign_data` input port to select dynamically between the two representations)
- Support for both signed and unsigned data representation formats
- Support for pipelining with parameterized output latency
- Active high asynchronous clear and clock enable control inputs

Common Applications

Accumulator applications include parity and magnitude detection, state machines, basic control logic, and other applications that involve determining data relationships.



For details about using an accumulator in parity detection applications, refer to *Using Parity to Detect Memory Errors in Stratix Devices White Paper*.

Resource Utilization & Performance

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

Table 1–1 summarizes the amount of resources required when using the altaccumulate function to implement an 8-bit unsigned accumulator.

Table 1–1. altaccumulate Megafunction Resource Usage

Device	Optimization	Width	Logic Usage
Stratix II, Stratix III	Speed	8 bits	8 ALUTs
	Balanced	8 bits	8 ALUTs
	Area	8 bits	8 ALUTs
Stratix, Stratix GX, HardCopy Stratix, Cyclone III, Cyclone II, Cyclone	Speed	8 bits	8 LEs
	Balanced	8 bits	8 LEs
	Area	8 bits	8 LEs

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 version 11.0 operating system, or a Sun workstation running the Solaris 8 or 9 operating system
- Quartus® II software version 6.0

MegaWizard Plug-In Manager Customization

You can use the MegaWizard® Plug-In Manager to set the altaccumulate megafunction ports and parameters for each accumulator in the design.

Search for “altaccumulate” in the Quartus II Help for a listing of the parameters to use when instantiating the megafunction if you are not using the MegaWizard Plug-In Manager.

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

- On the Tools menu, click the **MegaWizard Plug-In Manager** command.
- When working in the Block Editor, from the **Edit** menu, click **Insert Symbol as Block**, or:

Right-click in the Block Editor, point to **Insert**, and click **Symbol as Block**.

Then, in the Symbol dialog box, click **MegaWizard Plug-In Manager**.

- 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 altaccumulate megafunction. [Tables 2-1](#) and [2-2](#) show the features or settings for the altaccumulate megafunction. You can use these tables to determine appropriate settings for your accumulator designs.

Use Page 3 of the altaccumulate MegaWizard Plug-In Manager to select the data representation and specify the input data widths (refer to [Figure 2-1](#)).

Figure 2-1. MegaWizard Plug-In Manager - altaccumulate [Page 3 of 5]

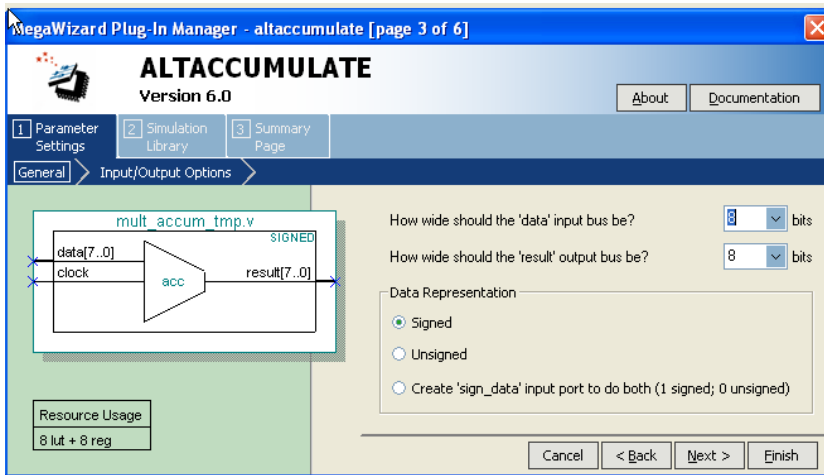


Table 2-1 shows the options available on Page 3 of the altaccumulate wizard.

Parameter	Description
How wide should the 'data input' bus be?	Select the width of the input data to be accumulated.
How wide should the 'result output' bus be?	Select the width of the output bus.
Data Representation	Choose whether you want the accumulation performed to be represented as signed or unsigned, or create a <code>sign_data</code> input port to dynamically select between the two representations.


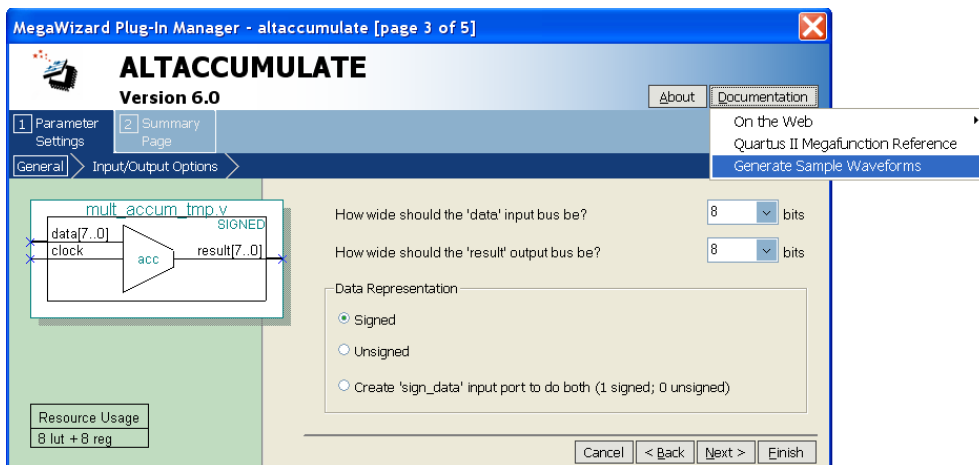
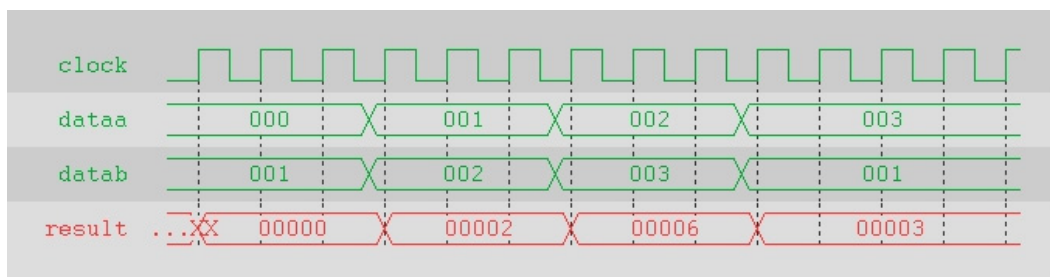
 Starting on Page 3 of the altaccumulate wizard, you can generate a sample simulation waveform and launch the Quartus II software Help for the altaccumulate megafunction by selecting **Generate Sample Waveforms** or **Quartus II Megafunction Reference** from the **Documentation** button (see [Figure 2-2](#)). Selecting **On the Web** brings up this user guide.

Figure 2–2. Documentation Button in the altaccumulate Wizard

The sample waveform as shown in [Figure 2–3](#) indicates the behavior of the altaccumulate megafunction for the chosen set of parameters in the altaccumulate design module. This option generates a sample waveform in HTML format in the specified design directory. The HTML file contains descriptions showing the accumulator operation.

Figure 2–3. Sample Waveforms for the altaccumulate Megafunction

On Page 4 of the altaccumulate wizard (see [Figure 2–4](#)) you can specify optional input and output ports for the accumulator.

Figure 2–4. MegaWizard Plug-In Manager - altaccumulate [Page 4 of 6]

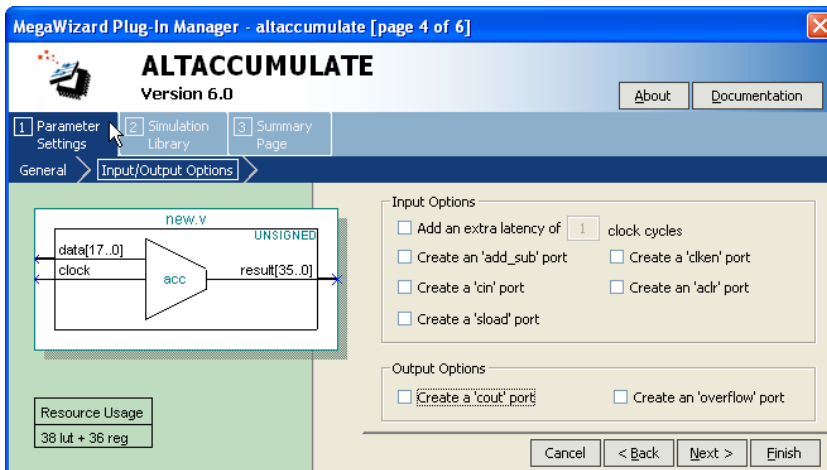


Table 2–2 shows the options available on Page 4 of the altaccumulate wizard. For more information on the ports, refer to Tables 3–1 and 3–2 in this user guide.

Table 2–2. altaccumulate Wizard Page 4

Parameter	Description
Input ports	Select which input ports you would like to add: <ul style="list-style-type: none"> ● Add an extra latency of (specify how many clock cycles) clock cycles ● Create an 'add_sub' port ● Create a 'cin' port ● Create a 'sload' port ● Create a 'clken' port ● Create an 'aclr' port
Output ports	<ul style="list-style-type: none"> ● Create a 'cout' port ● Create an 'overflow' port

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. That is, the Quartus II software uses the Altera megafunction code when compiling your design even though you did not specifically instantiate the megafunction. The Quartus II software infers megafunctions because they are optimized for Altera devices, so the area use and/or performance may be better than generic HDL code. Additionally, you must use megafunctions to access certain Altera architecture-specific features (for example, memory, DSP blocks, and shift registers) that improve performance compared with basic LEs.



Refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook* for 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 (a clear-box methodology). Both clear-box and black-box methodologies are described in 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. You can locate your megafunction in the Project Navigator window by expanding the compilation hierarchy and locating the megafunction by its name.

Similarly, 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 Simulation tool provides an integrated solution for performing simulations. The following sections describe the simulation options.

Quartus II Simulation

With the Quartus II Simulator, you can perform functional and timing simulation. A functional simulation in Quartus II 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 your RTL code. When performing a functional simulation, you add only

signals that exist before synthesis. You can find these signals with the **Registers: pre-synthesis, Design Entry**, or the pin filters in the Node Finder. The top-level ports are found using these filters.

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, you 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 process, the names of your RTL signals will change. Therefore, it might be difficult to find signals from your megafunction instantiation in the **Post-Compilation** filter. However, if you want 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 & Synthesis to keep a particular wire, register, or node intact. More information on these attributes is available in the *Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

EDA Simulation

Depending on the third-party simulation tool you are using, refer to the appropriate chapter in the *Simulation* section 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 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 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, 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 will then embed the analyzer along with your design seamlessly in the selected device.

For more information about using the SignalTap II embedded logic analyzer, refer to the *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Design Example: 8-Bit Multiplier Accumulator

Accumulators are common basic building blocks useful in DSP functions. This section presents a design example that uses the altaccumulate megafunction to generate a basic multiplier accumulator. 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 the 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

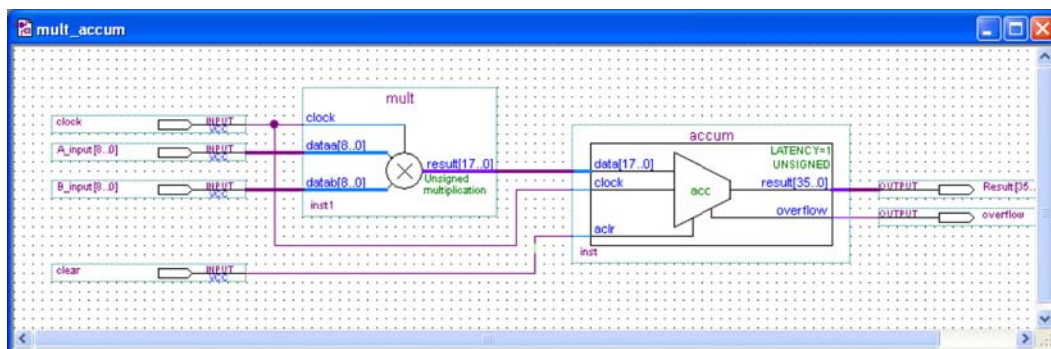
This example shows how to instantiate an altaccumulate megafunction using the MegaWizard Plug-In Manager. In this case, an altaccumulate is instantiated after a multiplier to create the multiplier accumulator. You can change the parameters as needed for your own design.

In this example, you will perform the following activities:

- Generate an 8-bit accumulator design module
- Implement the multiplier accumulator design module in an Altera device by assigning the Stratix II EP2S15F484C3 device and compiling the project
- Simulate the customized accumulator design module

Figure 2–5 shows the multiplier accumulator megafunction design.

Figure 2–5. Multiplier Accumulator Design

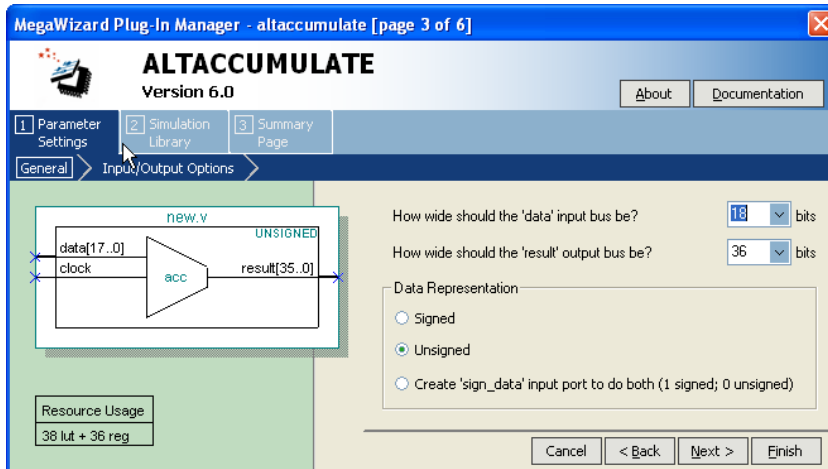


Generate an 8-Bit Accumulator

Use the following steps to generate an 8-bit accumulator.

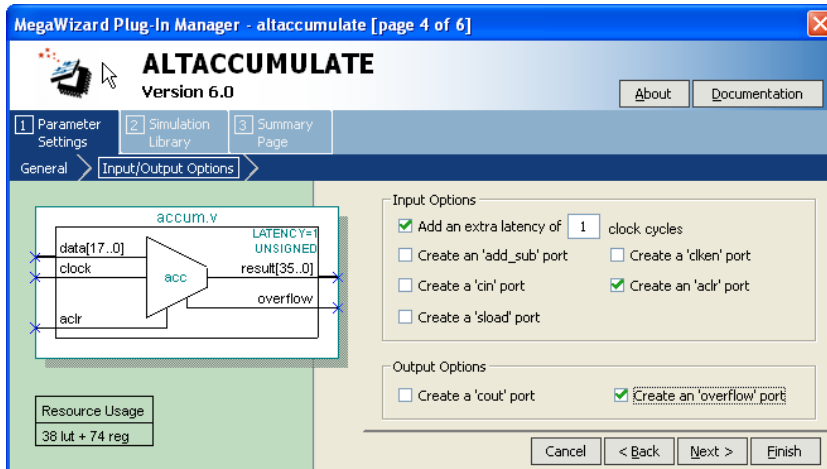
1. Open the project file **mult_accum.qpf**.
2. Open the top-level file **mult_accum.bdf**. This file is an incomplete design. You will be adding the accumulator to create the multiplier accumulator.
3. Double-click on a blank area in the block design (**.bdf**) file. In the **Symbol** window, or from the **Tools** menu, select **MegaWizard Plug-In Manager**.
4. On Page 1 of the MegaWizard Plug-In Manager, select **Create a new custom megafunction variation** and click **Next**. Page 2a appears.
5. On Page 2a, expand the Arithmetic folder and select the **ALTACCUMULATE** megafunction.
6. Select **Stratix II** for **Which device family will you be using?**.
7. Select **Verilog HDL** for **Which type of output file do you want to create?**.
8. Specify the output file name as *<project directory>\accum*.
9. Click **Next**. Page 3 is shown (Figure 2-6).

Figure 2–6. MegaWizard Plug-In Manager - altaccumulate [Page 3 of 6]



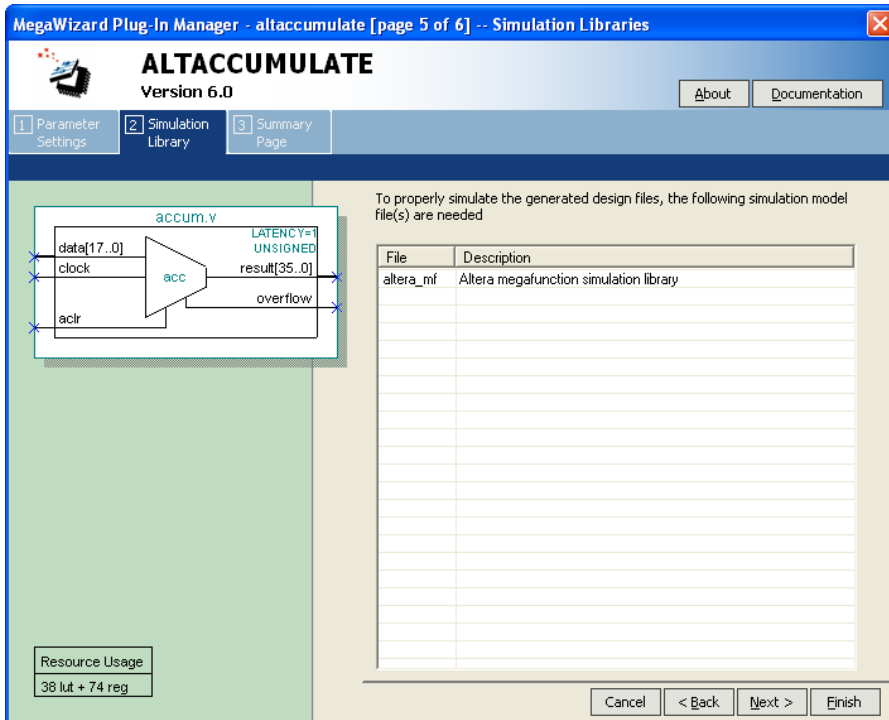
10. On Page 3 of the altaccumulate wizard, select **18** bits for **How wide should the 'data' input bus be?**
11. Select **36** bits for **How wide should the 'result' output bus be?**
12. Select **Unsigned** for **Data Representation**.
13. Click **Next**. Page 4 appears.

Figure 2–7. MegaWizard Plug-In Manager - altaccumulate [Page 4 of 6]



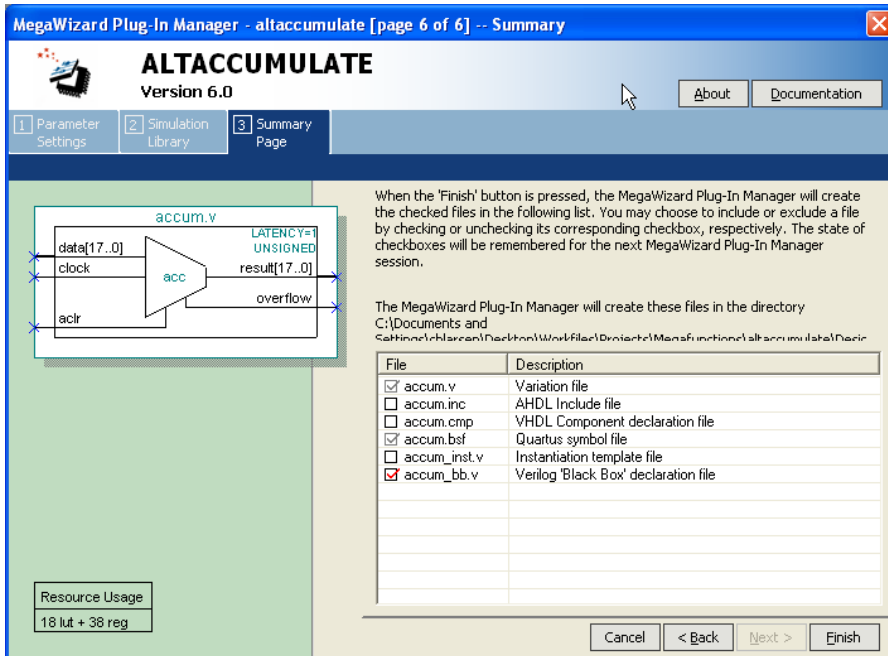
14. On Page 4, select **Add an extra latency of clock cycles** and set it to **1** clock cycle (Figure 2–7).
15. Select **Create an 'aclr' port**.
16. Select **Create an 'overflow' port**.
17. Click **Next**. Page 5 appears, listing the simulation model file(s) needed to properly simulate the generated design files (Figure 2–8).

Figure 2–8. MegaWizard Plug-In Manager - altaccumulate [Page 5 of 6]



18. Click Next. Page 6 appears.

Figure 2–9. MegaWizard Plug-In Manager - altaccumulate [Page 6 of 6]—Summary



19. Verify that the Quartus Symbol File (.bsf) is checked (Figure 2–9).
20. Click **Finish**. The altaccumulate megafunction is built.
21. Click **OK**.


Completing the Design File

1. Move the mouse to place the `accum` symbol in between the input and output ports of the `mult_accum.bdf` file.
2. Click the left mouse button to place the accumulator symbol.
3. Connect the output of the multiplier block (result port) to the data port of the accumulator, and similarly connect the clock and signals as shown in Figure 2–5.
4. From the **File** menu, select **Save**.

You have now completed the design file (Figure 2–5).

Implement the 8-Bit Multiplier-Accumulator

Compile the project to implement the 8-bit multiplier accumulator by performing the following steps:

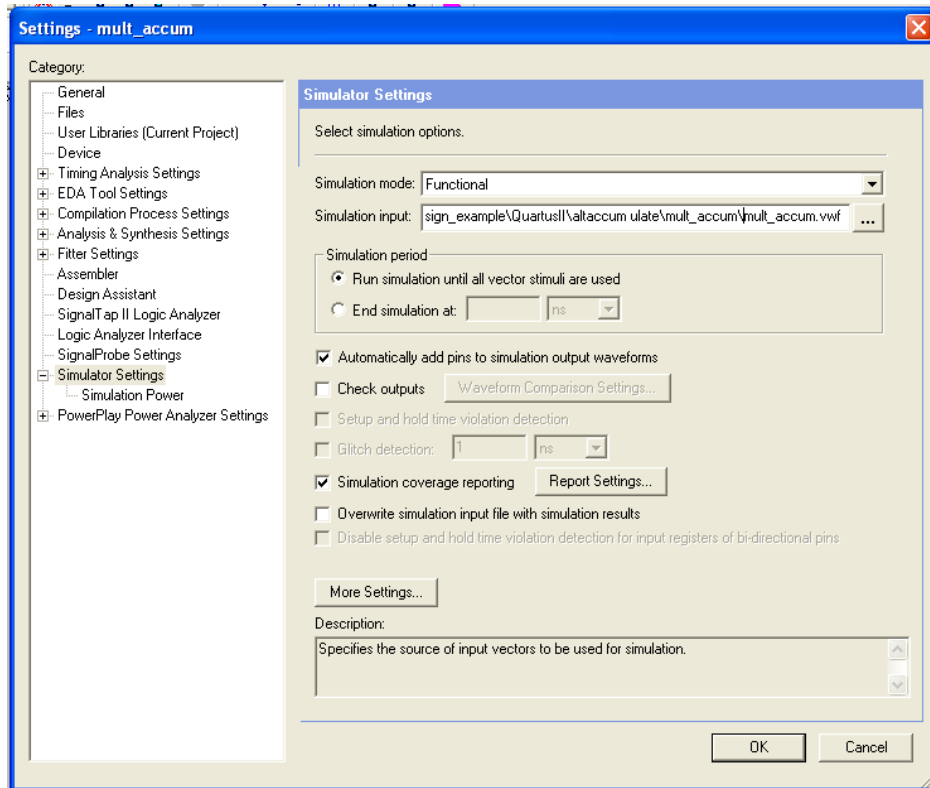
1. From the **Project** menu, select **Add/Remove Files in Project** Add **mult.v**, **accum.v**, and **mult_accum.bdf** to the project.
2. From the **Processing** menu, select **Start Compilation**, or click the compile button  to compile the design.
3. Click **Yes** to save any changes in the dialog box asking **Save changes to mult_accum.bdf?**
4. Click **OK** when the **Full compilation was successful** message box is shown.
5. To view how the module is implemented in the Stratix II device, from the **Assignments** menu, select **Timing Closure Floorplan**.

Simulate the 8-Bit Multiplier-Accumulator Design in Quartus II

Use the Quartus II software to simulate the multiplier accumulator design module and verify the results. Set up the Quartus II simulator by performing the following steps:

1. From the **Processing** menu, select **Generate Functional Simulation Netlist**.
2. When the **Functional simulation netlist generation was successful** message box appears, click **OK**.
3. From the **Assignments** menu, select **Settings** to open the Settings dialog box.
4. Select the **Simulator Settings** category in the Category list.
5. Under **Simulation mode**, select **Functional** and then select the **mult_accum.vwf** input vector waveform file (Figure 2-10).

Figure 2–10. Simulator Settings




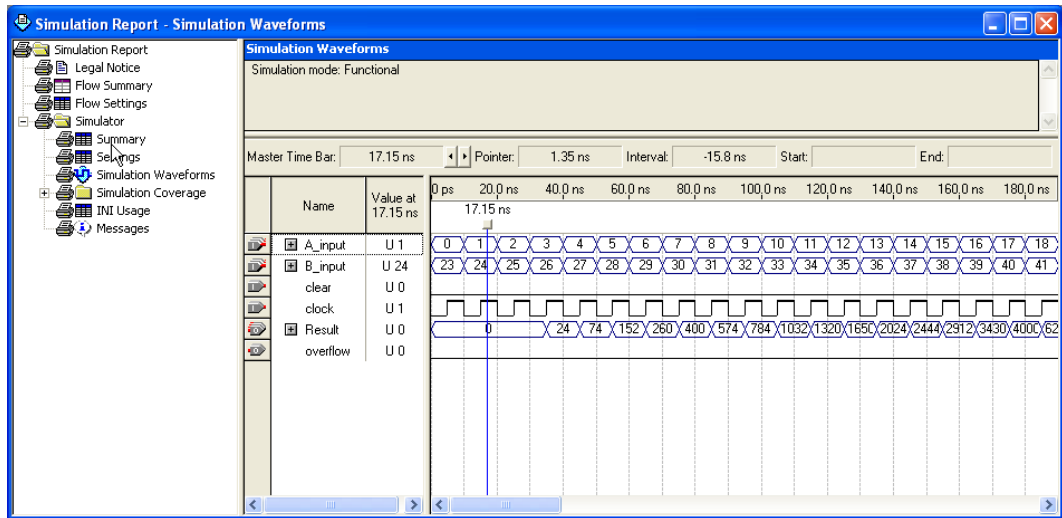
6. Click **Open** and then click **OK**.
7. From the **Processing** menu, select **Start Simulation**, or press Ctrl+I, or click on the **Simulation** button  to run a simulation.
8. When the simulation completes, a **Simulator was successful** message box appears. Click **OK**.
9. In the **Simulation Report** window, view the simulation output waveforms and verify the results. [Figure 2–11](#) shows the expected simulation results.

Figure 2–11. *mult_accum* Simulation Results

Simulate the 8-Bit Multiplier-Accumulator Design in ModelSim-Altera

Simulate the design in ModelSim to compare the results of both simulators. Note that this ModelSim design example is for the ModelSim-Altera (Verilog) version.

This User Guide assumes that you are familiar with using ModelSim-Altera before trying out the design example. If you are unfamiliar, please 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 here to topics such as installation, usage, and troubleshooting.

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

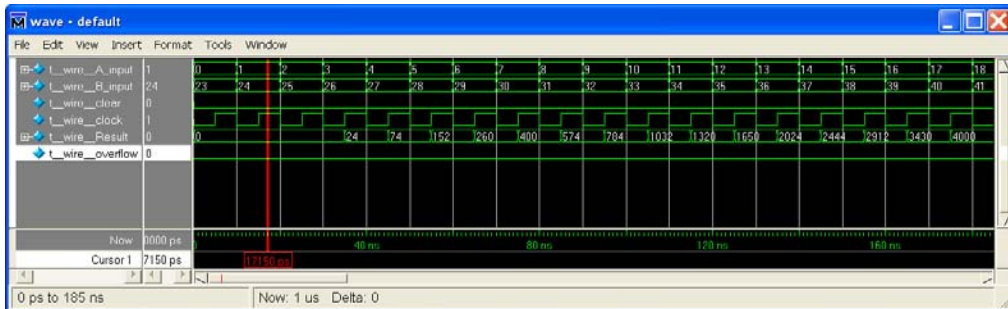
1. Unzip the **altaccumulate_msim.zip** file to any working directory on your PC.
2. Browse to where you have unzipped the files and open the **mult_accum.do** file in a text editor.

3. In line 1 of the `mult_accum.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. From the **File** menu, select **Save**.
5. Start **ModelSim-Altera**.
6. From the **File** menu, select **Change Directory**.
7. Select the folder where you have unzipped your files and click **OK**.
8. From the **Tools** menu, select **Execute Macro**.
9. Select the `mult_accum.do` file and click **Open**. This is a script file for ModelSim which automates all the necessary settings for the simulation.
10. Verify the results by looking at it in 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 2–12 shows the expected simulation results in ModelSim.

Figure 2–12. 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. The `altaccumulate` megafunction implements an accumulator, which stores the sum of a serial data stream. You can use the MegaWizard Plug-In Manager to parameterize this megafunction for specific needs.

Ports & Parameters

This chapter describes the ports and parameters for the altaccumulate megafunction.

Figure 3-1 shows all the available ports and parameters for an altaccumulate megafunction. Tables 3-1 and 3-2 show the input and output ports, respectively. Table 3-3 shows the parameterized megafunction interface of the altaccumulate megafunction.

These parameter details are relevant to users who do not use 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 if you use the MegaWizard Plug-In Manager interface.


 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. altaccumulate Symbol

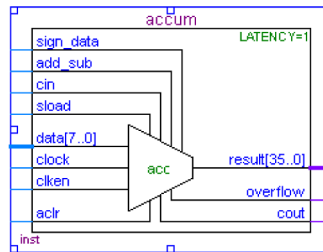


Table 3–1 shows the input ports for the altaccumulate megafunction.

Table 3–1. altaccumulate Megafunction Input Ports			
Port Name	Required	Description	Comments
<code>data[]</code>	Yes	Data input to the accumulator.	Input port [<code>WIDTH_IN-1..0</code>] wide.
<code>clock</code>	Yes	Clock input to the accumulator.	
<code>cin</code>	No	Carry-in to the low-order bit for add operations. Borrow-in to the low-order bit for subtract operations.	If omitted, the default is 0 for add operations and 1 for subtract operations.
<code>clken</code>	No	Clock enable for the <code>clock</code> port.	
<code>sload</code>	No	Synchronous load input. Loads the accumulator with the value of the <code>data[]</code> port on the next active clock edge.	Default = 0 (disabled). If the <code>sload</code> port is used, the <code>data[]</code> port must be connected.
<code>aclr</code>	No	Asynchronous clear input.	
<code>add_sub</code>	No	Controls the operation of the accumulator.	If the signal is high, the accumulator performs an add function. If the signal is low, the accumulator performs a subtract function. If this parameter is not used, the accumulator performs add functions only.
<code>sign_data</code>	No	Specifies the numerical representation of the <code>data[]</code> port.	If the <code>sign_data</code> port is high, the accumulator treats the <code>data[]</code> port as a signed two's complement. If the <code>sign_data</code> port is low, the accumulator treats the <code>data[]</code> port as an unsigned two's complement.

Table 3–2 shows the input ports for the altaccumulate megafunction.

Port Name	Required	Description	Comments
result[]	Yes	Accumulator output port.	Output port [WIDTH_OUT-1..0] wide
cout	No	Carry-out (borrow-in) of the 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	Overflow port for the accumulator.	—

Table 3–3 shows the parameterized megafunction interface of the altaccumulate megafunction.

Parameter	Type	Required	Comments
WIDTH_IN	Integer	Yes	Width of the data[] port.
WIDTH_OUT	Integer	Yes	Width of the result[] port.
LPM_REPRESENTATION	String	No	Specifies the numerical representation of the data[] input. Values are UNSIGNED and SIGNED. When this parameter is set to UNSIGNED, the accumulator interprets the data[] input as an unsigned two's complement. When this parameter is set to SIGNED, the accumulator interprets the data[] as a signed two's complement. This parameter should not be used if the sign_data port is specified. If omitted the default is UNSIGNED.
EXTRA_LATENCY	Integer	No	Specifies the number of clock cycles of latency.
USE_WYS	String	No	Specifies whether to construct an optimized accumulator with the data from the result[] port that cannot be merged with any other logic. Values are ON and OFF. If omitted, the default is ON.

