

lpm_rom Megafunction

User Guide



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

Software Version:	4.2
Document Version:	1.0
Document Date:	March 2005

Copyright © 2005 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-031105-1.0

Revision History	v
How to Contact Altera	v
Typographic Conventions	vi

Chapter 1. About These Megafunctions

Device Family Support	1-1
Introduction	1-2
Features	1-2
General Description	1-2
Stratix II and Stratix Device Families	1-3
Cyclone II Device Family	1-3
Cyclone Device Family	1-3
APEX II Device Family	1-4
Mercury Device Family	1-4
FLEX 10K Device Family	1-5
Memory Blocks	1-5
Common Applications	1-5
Resource Utilization & Performance	1-6

Chapter 2. Getting Started

System & Software Requirements	2-1
Mega Wizard Plug-In Manager Customization	2-1
MegaWizard Page Description	2-2
Inferring Megafunctions from HDL Code	2-6
Instantiating Megafunctions in HDL Code	2-6
Identifying a Megafunction after Compilation	2-7
Simulation	2-7
Quartus II Simulation	2-7
EDA Simulation	2-8
SignalTap II Embedded Logic Analyzer	2-8
Design Example: Single Port (32 x 8) Synchronous ROM	2-8
Design Files	2-8
Example	2-8
Generate a Single Port ROM	2-9
Implement the Single Port ROM	2-12
Functional Results - Simulate the Single Port ROM	2-13
Conclusion	2-14

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.

Chapter	Date	Document Version	Changes Made
All	March 2005	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/	www.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	literature@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 These Megafunctions

Device Family Support

Megafunctions provide either full or preliminary support for target Altera® device families, as described below:

- *Full support* means the megafunction meets all functional and timing requirements for the device family and may be used in production designs
- *Preliminary support* means the megafunction meets all functional requirements, but may still be undergoing timing analysis for the device family; it may be used in production designs with caution

Table 1–1 shows the level of support offered by the Altera `lpm_rom` megafunction for each Altera device family.

Table 1–1. Device Family Support	
Device Family	Support
Stratix® II	Preliminary
Stratix GX (1)	Full
Stratix (1)	Full
HardCopy™ Stratix (1)	Full
Cyclone™ II (1)	Preliminary
Cyclone (1)	Full
APEX™ II	Full
APEX 20K	Full
Mercury™	Full
FLEX™ 10K	Full
ACEX® 1K	Full
Excalibur	Full

Note to Table 1–1:

- (1) To implement all ROM functions, the `lpm_rom` function is provided only for backward compatibility in Cyclone series, Stratix, Stratix GX, and HardCopy Stratix device designs. Altera recommends using the `altsyncram` megafunction which is available for all Altera devices supported by the Quartus® II software except MAX 3000 and MAX 7000 devices.

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_rom` megafunction is a storage megafunction, implementing a single-port ROM function and offers many additional features, which include:

- Fully parameterizable
- Provides synchronous or asynchronous single-port ROM
- Support for read access to memory cells
- Registers EAB/ESB address inputs using `inclock`
- Registers output data using a separate `outclock`
- EAB/ESB or dedicated memory blocks based implementation
- ROM initialization using memory initialization file or HEX file
- Instantiates easily with the MegaWizard® Plug-In Manager

General Description

The `lpm_rom` megafunction is a single-port ROM megafunction with separate input and output ports. This megafunction supports both synchronous and asynchronous modes of operation. The `address []` input port and the `q []` output port can be registered and are controlled by the `inclock` and `outclock` inputs respectively. Totally asynchronous memory operations occur when both the `inclock` and `outclock` ports are unused.

The `lpm_rom` megafunction provides the optional use of a memory enable signal, the `memenab` port. When the memory is not enabled (`memenab` is low), the `q []` output is high-impedance.



`lpm_rom` must have a memory initialization file (**.mif**) or hexadecimal (**.hex**) file with the same name in the project directory that contains the data written to ROM during configuration.

Use the `lpm_rom` megafunction to read data from, and write data to, in-system memory in devices with the **In-System Memory Content Editor**.

ROM is implemented in Altera devices with the following embedded structures (Table 1-2):

Table 1–2. Embedded Memory Blocks in Altera Devices	
Type	Device Family
Embedded Array Blocks (EABs)	FLEX 10K and ACEX 1K families
Embedded System Blocks (ESBs)	APEX II, APEX 20K, Mercury, and Excalibur devices
M512 memory blocks	Stratix II, Stratix GX, Stratix, and HardCopy Stratix device families
M4K memory blocks	Cyclone series, Stratix series, and HardCopy Stratix device families

Stratix II and Stratix Device Families

In Stratix and Stratix II devices, both M512 and the M4K embedded memory blocks support ROM mode. The M-RAM block does not support ROM mode. Stratix device memory configuration must have synchronous inputs. Therefore, the address lines of the ROM are registered. The outputs can be registered or combinatorial. The ROM read operation is identical to the read operation in the single-port RAM configuration.



For more information, refer to the *Memory* section in volume 2 of both the *Stratix II Device Handbook* and the *Stratix Device Handbook*.

Cyclone II Device Family

Cyclone II device memory blocks are arranged in columns across the device between specific LABs. Cyclone II devices offer between 119 to 1,152 Kbits of embedded memory. M4K memory blocks are true dual-port memory blocks with 4K bits of memory plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 36-bits wide at up to 250 MHz.



For more information, refer to the *Memory* section in volume 1 of the *Cyclone II Device Handbook*.

Cyclone Device Family

In Cyclone devices, memory blocks are grouped into columns across the device in between certain LABs. Cyclone devices offer between 60 to 288 Kbits of embedded memory. M4K RAM blocks are true dual-port memory blocks with 4K bits of memory plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 36-bits wide at up to 200 MHz.



For more information, refer to the *On-Chip Memory Implementations Using Cyclone Memory Blocks* chapter in volume 1 of the *Cyclone Device Handbook*.

APEX II Device Family

In APEX II devices, ESBs implement many types of memory including Dual-Port+ RAM, CAM, ROM, and FIFO functions. The ESB includes input and output registers: input registers synchronize writes, and output registers pipeline designs to improve system performance. The ESB offers a bidirectional, dual-port mode, supporting any combination of two port operations: two reads, two writes, or one read and one write at two different clock frequencies.

Embedding the memory directly into the die improves performance and reduces die area compared to distributed-RAM implementations. The abundance of cascadable ESBs ensures that the APEX II device can implement multiple wide memory blocks for high-density designs. The high speed of the ESB ensures it can implement small memory blocks without any speed penalty. The abundance of ESBs, in conjunction with the ability for one ESB to implement two separate memory blocks, ensures that designers can create as many different-sized memory blocks as the system requires.



For more information, refer to the *APEX II Programmable Logic Device Family* data sheet.

Mercury Device Family

The Mercury architecture contains a row-based logic array to implement general logic and a row-based embedded system array to implement memory and specialized logic functions. The ESB can implement various types of memory blocks, including quadport, true dual-port, dual- and single-port RAM, ROM, FIFO, and CAM blocks.

The ESB includes input and output registers; the input registers synchronize reads and/or writes, and the output registers can pipeline designs to further increase system performance. The ESB offers a quad port mode, which supports up to four port operations, two reads and two writes simultaneously, with the ability for a different clock on each of the four ports.



For more information, refer to the *Mercury Programmable Logic Device Family* data sheet.

FLEX 10K Device Family

Each FLEX 10K device contains an embedded array to implement memory and specialized logic functions, and a logic array to implement general logic.

The embedded array consists of a series of EABs. When implementing memory functions, each EAB provides 2,048 bits, which can be used to create RAM, ROM, dual-port RAM, or first-in first-out (FIFO) functions. EABs can be used independently, or multiple EABs can be cascaded to expand the memory required.



For more information, refer to the *FLEX 10K Embedded Programmable Logic Device Family* data sheet.

Memory Blocks

An ASCII text file (**.mif**) specifying the initial content of a memory block (CAM, RAM, or ROM), that is, the initial values for each address, is used during project compilation and/or simulation.

A MIF is used as an input file for memory initialization in the Compiler and Simulator. You can also use a hexadecimal (Intel-format) file (**.hex**) to provide memory initialization data.

A MIF contains the initial values for each address in the memory. A separate file is required for each memory block. In a MIF, you are required to specify the memory depth and width values. In addition, you can specify the radices used to display and interpret addresses and data values.



If multiple values are specified for the same address, only the last value is used.

Common Applications

The `lpm_rom` megafunction is used in applications requiring parallel data transfer. Synchronous read operations into the memory block use the `address[]` which are triggered by the rising edge of the `inclock`. The `outclock` port is for the read operation.

Resource Utilization & Performance

The Quartus II Compiler automatically implements this function in embedded system blocks (ESB) in APEX 20K, APEX II, Excalibur, and Mercury devices, and in embedded array blocks (EAB) in ACEX 1K and FLEX10K devices. The Compiler automatically implements this function in logic cells in FLEX 6000 devices. It is unnecessary to use the auto implement in ROM logic option for this function.



Using the auto implement in ROM logic option for the `lpm_rom` function may cause warning messages to appear.

Table 1–3 summarizes the resource usage of the `lpm_rom` megafunction.

Table 1–3. <i>lpm_rom</i> Megafunction Resource Usage							
Device family	Optimization (1)	Width	Memory Bits	RAM block bits	RAM block type (2)	RAM block type used	Number of RAM blocks
Stratix II	Balanced	8-bit	256	576	Auto	M512	1
	Balanced	8-bit	256	4608	M4K	M4K	1
Stratix GX	Balanced	8-bit	256	576	Auto	M512	1
	Balanced	8-bit	256	4608	M4K	M4K	1
Stratix	Balanced	8-bit	256	576	Auto	M512	1
	Balanced	8-bit	256	4608	M4K	M4K	1
HardCopy Stratix	Balanced	8-bit	256	576	Auto	M512	1
	Balanced	8-bit	256	4608	M4K	M4K	1
Cyclone II	Balanced	8-bit	256	4608	Auto	M4K	1
Cyclone	Balanced	8-bit	256	4608	Auto	M4K	1
APEX II	Balanced	8-bit	256	4096	Auto	ESB	1
APEX 20K	Balanced	8-bit	256	2048	Auto	ESB	1
Mercury	Balanced	8-bit	256	4096	Auto	ESB	1
FLEX 10K	Balanced	8-bit	256	2048	Auto	EAB	1
ACEX 1K	Balanced	8-bit	256	4096	Auto	EAB	1
Excalibur	Balanced	8-bit	256	2048	Auto	ESB	1

Notes to Table 1–3:

- (1) Choose a design implementation that balances high performance with minimal logic usage. This setting is available for APEX 20K, Cyclone, Cyclone II, MAX[®]II, Stratix, and Stratix II devices only. The balanced optimization logic option is set in **Analysis and Synthesis settings** (Assignments menu).
- (2) Specify the RAM block type on page 5 of the MegaWizard Plug-In Manager of the `lpm_rom` megafunction.

System & Software Requirements

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

- A PC running Windows NT/2000/XP, Red Hat Linux 7.3 or 8.0, or 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 beginning with version 4.2

Mega Wizard Plug-In Manager Customization

Use the MegaWizard® Plug-In Manager to create or modify design files containing custom megafunction variations which can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard to set the `lpm_rom` megafunction features in the design.

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

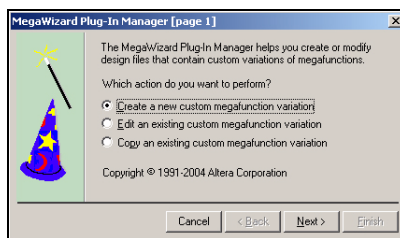
- Choose the **MegaWizard Plug-In Manager** command (Tools menu).
- When working in the Block Editor, click **MegaWizard Plug-In Manager** in the **Symbol** dialog box (Edit menu).
- Start the stand-alone version of the **MegaWizard Plug-In Manager** by typing the following command at the command prompt:
`qmegawiz` ↵

MegaWizard Page Description

This section provides descriptions of the options available on the individual pages of the `lpm_rom` MegaWizard Plug-In Manager.

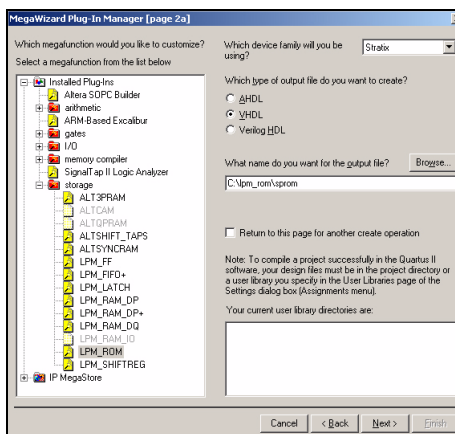
In page 1 of the MegaWizard Plug-In Manager, you can create, edit, or copy a custom megafunction variation (Figure 2-1).

Figure 2-1. Create a New Megafunction Variation

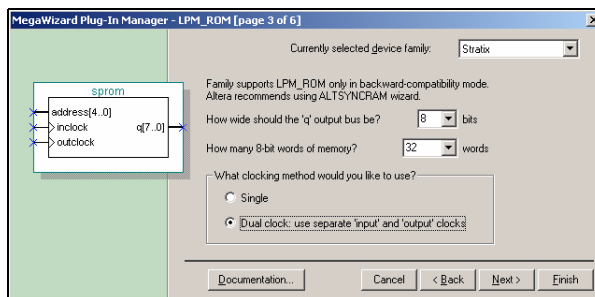


On page 2a, specify the family of device you want to use, type of output file to create, and the name of the output file. You can choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v) as the output file type (Figure 2-2).

Figure 2-2. MegaWizard Plug-In Manager



On page 3 of the wizard, you specify customizable parameters for the device family, data bus width, the type of reset, and the clock enable option (Figure 2-3 on page 2-3). A description of each feature is shown in Table 2-1.

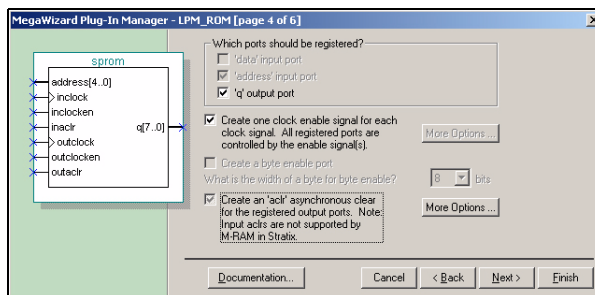
Figure 2–3. *lpm_rom Wizard, Page 3***Table 2–1. *lpm_rom MegaWizard Plug-in Manager Page 3 Options***

Function	Description
Currently selected device family	Specify the Altera device family you are using.
How wide should the 'q' output bus be?	Select the width for the 'q' output bus. The maximum size of the 'q' output bus can be 256.
How many 8 bit words of memory?	Select the depth of the ROM in terms of the number of words. The no of bits in the word depends on the size of the 'q' output bus.
Which clocking method do you want to use? (1)	Select the clocking mode either single clock or dual clock. Dual clock mode, features separate 'input' and 'output' clocks

Note for Table 2–1:

(1) This feature is available for Stratix series and HardCopy Stratix devices only.

On page 4 of the wizard, you specify which port should be registered and create a clock enable for the clock signals (Figure 2–4). A description of each feature is shown in Table 2–2.

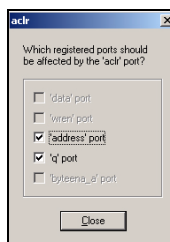
Figure 2-4. lpm_rom Wizard, Page 4**Table 2-2. lpm_rom MegaWizard Plug-in Manager Page 4 Options**

Function	Description
Which ports should be registered? (1)	Use this feature to register either 'address' input port or 'q' output port.
Create one clock enable signal for each clock signal. All registered ports are controlled by the enable signal(s).	Use this feature to create a clock enable for each of the clock signal. 'inclock' and 'outclock' have separate clock enable signals.
Create an 'aclr' asynchronous clear for the registered ports. (2)	Use this feature to asynchronously clear all the registered ports. More options allows to separately turn on aclr for 'address' input port and 'q' output port. (3)

Note for Table 2-2:

- (1) In Stratix series, HardCopy Stratix, and Cyclone series device families, option to register 'address' input port is unavailable. In APEX II, APEX 20K, ACEX 1K, FLEX 10K, and Excalibur device families, option to register both 'address' input port as well 'q' output port is available.
- (2) Input acls are not supported by M-RAM in Stratix devices.
- (3) See Figure 2-5.

To asynchronously clear all the registered ports, use the **Create an 'aclr' asynchronous clear for the registered ports** option (Figure 2-5).

Figure 2-5. Asynchronous Clear options

On page 5 of the `lpm_rom` wizard, specify the initial content of memory and type of RAM block (Figure 2–6). A description of each feature is shown in Table 2–3.

Figure 2–6. `lpm_rom` Wizard, page 5

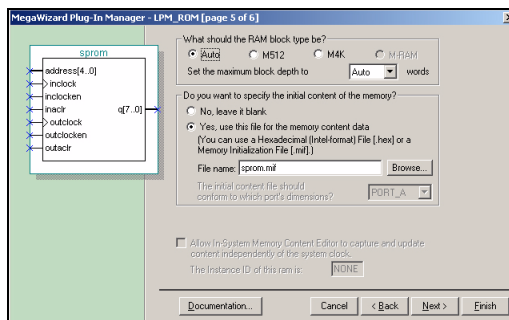


Table 2–3. `lpm_rom` MegaWizard Plug-in Manager Page 5 Options

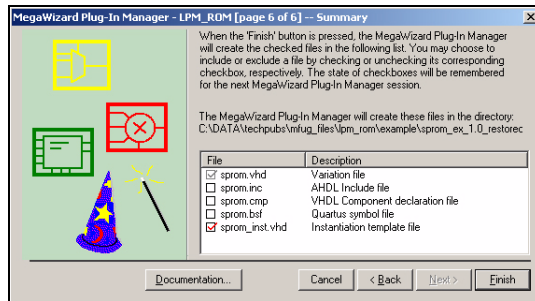
Function	Description
What should the RAM block type be?	Specify which RAM block type to be used. <ul style="list-style-type: none"> • Select 'Auto' to allow the compiler to implement the ROM in suitable memory blocks • Select either 'M512' or 'M4K' block type to choose a specific embedded memory type for ROM implementation.
Set the maximum block depth to	Fix up the block depth to any value lesser than the maximum depth of the block. The compiler assigns the number of memory blocks to be used for a given size of memory implementation. <ul style="list-style-type: none"> • Select 'Auto' to allow the compiler to take the maximum depth of the memory block.
Do you want to specify the initial content of the memory?	Select 'No, leave it blank' if you do not want to specify any initialization file. Select the second option to specify the initialization file. Type the file name or Browse for the required file.
Allow In-System Memory Content Editor to capture and update the content independently of the system clock. (1)	This feature is available only for Stratix and Cyclone devices where you want use the In-System Memory Content Editor.

Note to Table 2–3:

(1) This feature is available only for Stratix series and Cyclone series device families.

On page 6 of the wizard, specify the types of files to have generated for your custom megafunction. The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (indicated by a red check mark) (Figure 2–7 on page 2–6).

Figure 2–7. *lpm_rom Wizard, Page 6*



For more information on the ports for this megafunction, see the *Specifications* chapter in this User Guide.

Inferring Megafunctions from HDL Code

Synthesis tools, including the Quartus II software 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 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. These features generally provide improved performance compared to basic logic elements.



See the *Recommended HDL Coding Styles* chapter in 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 the third-party synthesis support chapters in the *Synthesis* section 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. You can locate your megafunction in the **Project Navigator** window by expanding the compilation hierarchy and locating the megafunction by its name.

To search for node names within the megafunction using **Node Finder**, click **Browse (...)** under **Look in**, 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 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 **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, 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, 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 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. More information on these attributes is available in the *Quartus II 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*. These 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 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 project. The Quartus II software will then seamlessly embed the analyzer along with your design in the selected device.



For more information on 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: Single Port (32 x 8) Synchronous ROM

This design example uses the `lpm_rom` megafunction to implement Single port 8-bit synchronous ROM. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described in detail. The new megafunction created in this example is added to the top-level file in your Quartus II project.

Design Files

The design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site:
<http://www.altera.com/support/examples/quartus/quartus.html>

Select the [Examples for lpm_rom Megafunction User Guide](#) link from the examples page to download the design files.

Example

In this example, you perform the following activities:

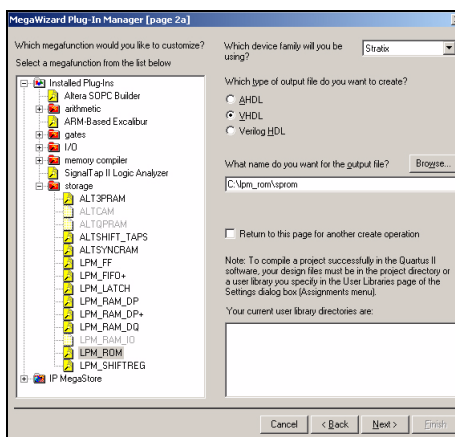
- Create a single port ROM using the `lpm_rom` megafunction and the MegaWizard Plug-in Manager

- Implement the design and assign the EP1S10F484C5 device to the project
- Compile and simulate the design

Generate a Single Port ROM

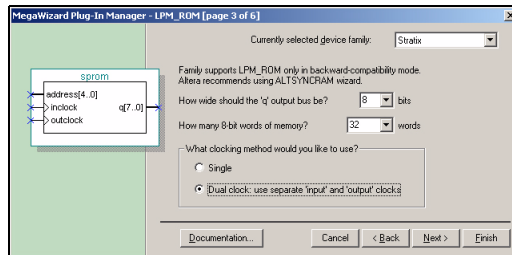
1. In the Quartus II software, open the **sprom.qar** project.
2. Select **MegaWizard Plug-In Manager** (Tools menu). Select **Create a new custom megafunction variation**, and click **Next**. The **MegaWizard Plug-In Manager** page displays (Figure 2–8).

Figure 2–8. MegaWizard Plug-In Manager



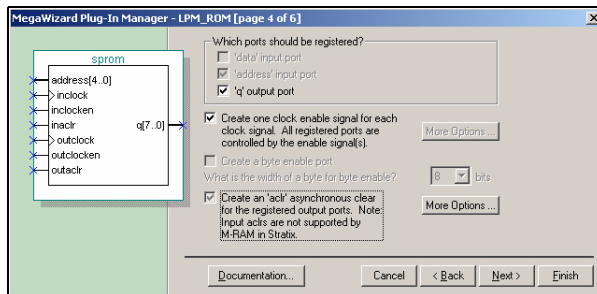
3. From **Which device family will you be using?**, select **Stratix**.
4. Under **Which type of output file do you want to create?**, click **Verilog HDL**.
5. In the **storage** folder, select **LPM_ROM**. Specify the output file **sprom**.
6. Click **Next**. Figure 2–9 on page 2–10 shows the wizard after you have made these selections.

Figure 2–9. lpm_rom Wizard, Page 3

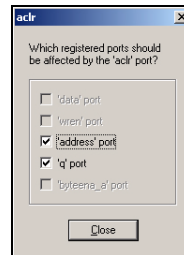


7. In the **How wide should the 'q' output bus be?** list, select 8.
8. In the **How many 8-bit words of memory?** list, select 32.
9. Under **What clocking method would you like to use?** option, click **Dual clock: use separate 'input' and 'output' clocks**.
10. Click **Next**. [Figure 2–10](#) shows the wizard after you have made these selections.

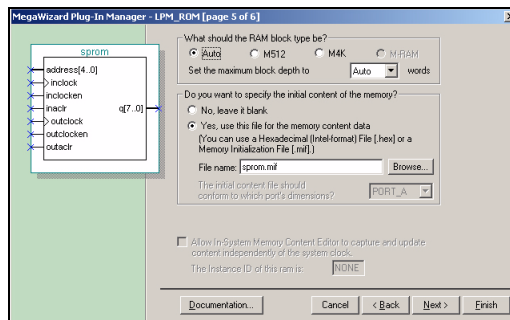
Figure 2–10. lpm_rom Wizard, Page 4



11. Under **Which ports should be registered?**, turn on **'q' output port**.
12. Turn on the **Create one clock enable signal for each clock signal** and **Create an 'aclr' asynchronous clear for registered output ports** options. Click **More Options** ([Figure 2–11](#) on [page 2–11](#)).

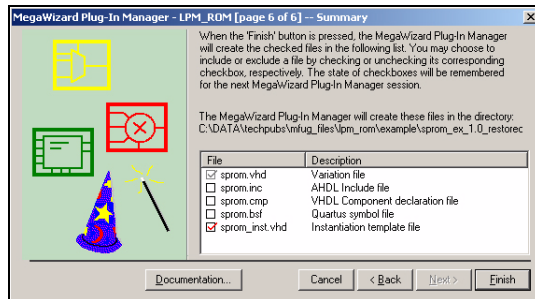
Figure 2–11. Asynchronous Clear Options

13. Under **Which registered ports should be affected by the 'aclr' port?**, turn on 'address' port and 'q' port.
14. Click **Close**. Page 4 displays.
15. Click **Next**. Figure 2–12 shows the wizard after you have made these selections.

Figure 2–12. lpm_rom Wizard, Page 5

16. Under **What should the RAM block type be?**, click **Auto**, and select **Auto** from the **Set the maximum block depth to** option.
17. Under **Do you want to specify the initial content of the memory?**, click **Yes, use this file for the memory content data**.
18. In **File name**, enter **sprom.mif**.
19. Click **Next**. Figure 2–13 on page 2–12 shows the wizard summary.

Figure 2–13. *lpm_rom Wizard, Summary*



20. Turn on **Instantiation template file**.
21. Turn off **Quartus Symbol file**, **AHDL include file**, and **VHDL Component declaration file**. Click **Finish**.

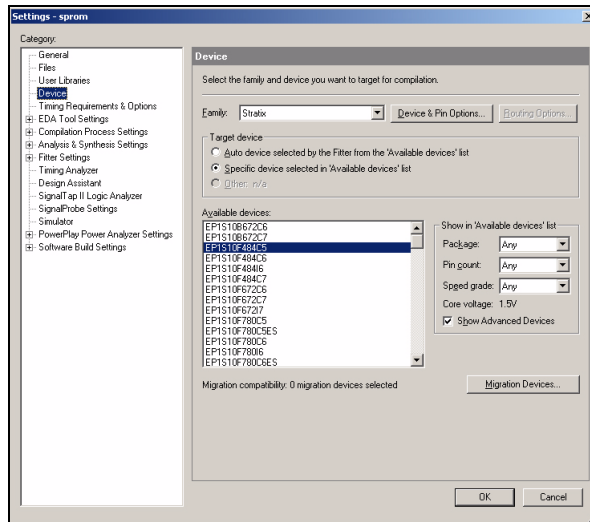
The `lpm_rom` module is now built.

Implement the Single Port ROM

This section describes how to assign the EP1S10F484C5 device to the project and how to compile the project.

1. In the Quartus II software, select **Settings** (Assignments menu). The **Settings** window displays.
2. Select **Device** from the Category list. The **Device Selection** window displays (Figure 2–14).

Figure 2–14. Device Selection



3. From **Which device family will you be using?**, select **Stratix**.
4. Under **Target device**, click **Specific device selected in 'Available devices' list**.
5. Under **Available devices**, select **EP1S10F484C5**.
6. Click **OK**.
7. Choose **Start Compilation** (Processing menu).
8. The **Full compilation was successful** box displays. Click **OK**.

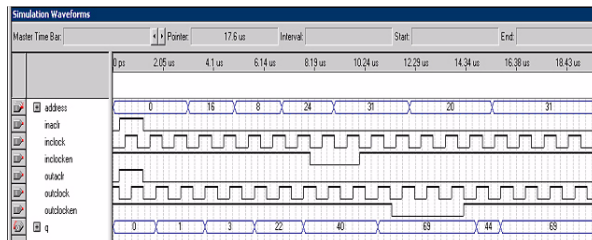
Functional Results - Simulate the Single Port ROM

This section describes how to verify the design example you just created by simulating the design using the Quartus II Simulator. To set up the Quartus II Simulator, perform the following steps:

1. From the Quartus II software, choose **Generate Functional Simulation Netlist** (Processing menu).
2. The **Functional Simulation Netlist Generation was successful** box displays. Click **OK**.

3. Choose **Settings** (Assignments menu). The **Settings** dialog box displays.
4. In the **Category** list, select **Simulator**.
5. In the **Simulation mode** list, select **Functional**.
6. Type **sprom_ip.vwf** in the **Simulation input** box, or click **Browse (...)** to select the file in the project folder.
7. Turn on the **Run simulation until all vector stimuli are used**, **Automatically add pins to simulation output waveforms**, and **Simulation coverage reporting** options.
8. Turn off **Overwrite simulation input file with simulation results**.
9. Click **OK**.
10. Choose **Start Simulation** (Processing menu).
11. The **Simulation was successful** box displays. Click **OK**.
12. The **Simulation Report** window displays. Verify the simulation waveform results (Figure 2–15).

Figure 2–15. Simulation Waveforms



Conclusion

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

Ports & Parameters

This chapter describes the ports and parameters for the `lpm_rom` megafunction.

The parameter details are only relevant for users who by-pass 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 Help for the most current information on the ports and parameters for these megafunctions.

Figure 3–1 shows the ports and parameters for the `lpm_rom` megafunction.

Figure 3–1. `lpm_rom` Ports and Parameters

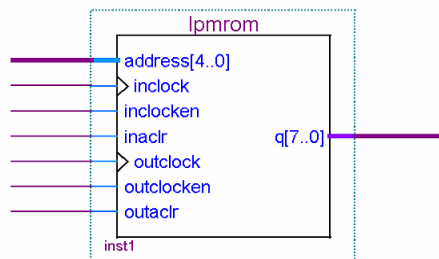


Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the `lpm_rom` megafunction parameters.

Table 3–1. <i>lpm_rom</i> Input Ports			
Name	Required	Description	Comment
<code>address[]</code>	Yes	Address input to the memory	Input port <code>LPM_WIDTHAD</code> wide.
<code>inclock</code>	No	Clock for input registers.	The <code>address[]</code> port is synchronous (registered) when <code>inclock</code> port is connected, and is asynchronous (unregistered) when the <code>inclock</code> port is not connected.
<code>outclock</code>	No	Clock for output registers.	Addressed memory content-to- <code>q[]</code> response is synchronous when <code>outclock</code> port is connected, and is asynchronous when it is not connected.
<code>Memenable</code>	No	Memory enable input.	High = data output on <code>q[]</code> , Low = high-impedance outputs. (1)

Note to Table 3–1:

- (1) This port is available for backward compatibility only and Altera® recommends that you *not* use this port.

Table 3–2. <i>lpm_rom</i> Output Ports		
Name	Required	Description
<code>q[]</code>	Yes	Output of memory.

Table 3–3. <i>lpm_rom</i> Parameters (Part 1 of 2)			
Parameter	Type	Required	Description
<code>LPM_WIDTH</code>	Integer	Yes	Width of the <code>q[]</code> port.
<code>LPM_WIDTHAD</code>	Integer	Yes	Width of the <code>address[]</code> port. <code>LPM_WIDTHAD</code> should be (but is not required to be) equal to $\text{LOG2}(\text{LPM_NUMWORDS})$. If <code>LPM_WIDTHAD</code> is too small, some memory locations will not be addressable. If it is too large, the addresses that are too high returns undefined logic levels.
<code>LPM_NUMWORDS</code>	Integer	No	Number of words stored in memory. In general, this value should be (but is not required to be) $2^{\text{LPM_WIDTHAD}} - 1 < \text{LPM_NUMWORDS} \leq 2^{\text{LPM_WIDTHAD}}$. If omitted, the default is $2^{\text{LPM_WIDTHAD}}$.
<code>LPM_FILE</code>	String	Yes	Name of the Memory Initialization File (<code>.mif</code>) or Hexadecimal (Intel-Format) Output File (<code>.hexout</code>) containing ROM initialization data ('<file name>'), or 'UNUSED'.
<code>LPM_ADDRESS_CONTROL</code>	String	No	Values are 'REGISTERED', 'UNREGISTERED', and 'UNUSED'. Indicates whether the address port is registered. If omitted, the default is 'REGISTERED'.

Table 3–3. *lpm_rom* Parameters (Part 2 of 2)

Parameter	Type	Required	Description
LPM_OUTDATA	String	No	Values are 'REGISTERED', 'UNREGISTERED', and 'UNUSED'. Indicates whether the <i>q</i> and <i>eq</i> ports are registered. If omitted, the default is 'REGISTERED'.
LPM_HINT	String	No	Allows you to specify 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.

