RAM Megafunction

User Guide

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

Document Version: 1.0
Document Date: September 2004

UG-MF9404-1.0

Contents

Chapter 3. Specifications

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
1	September 2004	1.0	• Initial release
2	September 2004	1.0	• Initial release
3	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)	(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: $\mathbf{f_{MAX}}$, **\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.
	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 RAM megafunctions to each Altera device family.

Table 1–1. Device Family Support

Device Family	Support
Stratix® II	Full
Stratix	Full
Stratix GX	Full
HardCopy Stratix	Full
Cyclone™	Full
Mercury™	Full
Excalibur™	Full
ACEX® 1K	Full
APEX™ II	Full
APEX 20KE & APEX 20KC	Full
APEX 20K	Full
HardCopy™ APEX	Full
FLEX 10KE®	Full
FLEX 6000®	Full
MAX 3000® & MAX 7000®	Full
Other device families	No support

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.

The Quartus® II software provides three megafunctions that support single-port and dual-port RAM functionality: `lpm_ram_dp`, `lpm_ram_dq`, and `lpm_ram_io`. This chapter describes the features, descriptions, and resource usage of the RAM megafunctions.

Features of lpm_ram_dp

The `lpm_ram_dp` megafunction implements a dual-port RAM function and offers many additional features, which include:

- Fully parameterizable
- Support for simultaneous read and write access to memory cells
- Registers any combination of EAB/ESB inputs
- Is LE-based or EAB/ESB-based
- Provides four clock modes: Single, Shared, Separate, and Asynchronous

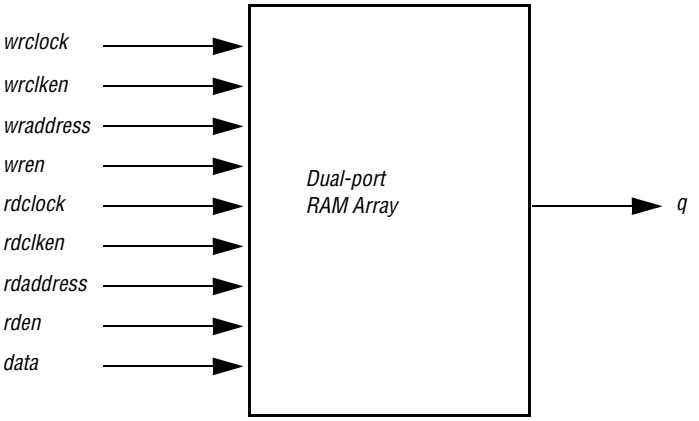
General Description of lpm_ram_dp

The `lpm_ram_dp` megafunction is a dual-port RAM megafunction provided in the Quartus II software MegaWizard® Plug-In Manager.

The `lpm_ram_dp` megafunction supports applications that require parallel data transfer in which two independent clock ports use different access rates for read and write operations. The presence of dual-addressing supports simultaneous read and write operations in the same clock cycle.

The `lpm_ram_dp` megafunction provides four modes of operation: single clock, shared clock, separate clock, and asynchronous. In single clock mode, the read and write operations are synchronous with the same clock. In shared clock mode, the read and write operations are synchronous with the same clock, and there is a separate clock for the output port, `q[]`. This is also referred to as separate input and output clocks. In separate clock mode, there are two independent clocks, `rdclock` (read clock) and `wrclock` (write clock), for the read/write operations, respectively. In asynchronous mode, no clock is required. The write operation is dependent only on the `wren` (write enable) signal. The read operation is dependent on the `rden` (read enable) signal. If not present, the `rden` signal is connected to VCC by default.

Figure 1–1. lpm_ram_dp Megafunction Block Diagram



Common Applications of lpm_ram_dp

The `lpm_ram_dp` megafunction is used to implement a dual-port memory in device families such as APEX 20K, APEX II, ARM-based Excalibur, Mercury, ACEX 1K, FLEX 10KE, FLEX 6000, MAX 3000, and MAX 7000. For other devices, Altera recommends using the `altsyncram` function to implement a dual-port memory.

Resource Utilization & Performance of lpm_ram_dp

The `lpm_ram_dp` megafunction uses the following device resources:

- Embedded System Blocks (ESB) in APEX 20K, APEX II, Excalibur, and Mercury devices
- Embedded Array Blocks (EAB) in ACEX 1K and FLEX 10KE devices
- DFFE primitives or latch arrays in FLEX 6000, MAX 3000, and MAX 7000 devices or if the USE_EAB parameter is set to "`OFF`"

The dual-port memory is implemented in LEs or ESBs (EABs) by enabling the **Implement with logic cells only, even if the device contains EABs or ESBs** option on page 3 of the megafunction wizard. When this option is turned on, the memory is implemented in LEs, even when the device contains other memory resources such as EABs or ESBs.

If the output port is registered, the performance of the dual-port memory is improved, because the output port is then pipelined and f_{MAX} is improved.

Features of lpm_ram_dq

The lpm_ram_dq megafunction implements a single-port RAM function and offers many additional features, which include:

- Provides synchronous or asynchronous single-port RAM
- Registers EAB/ESB data and/or address inputs using inclock
- Registers output data using a separate outclock
- LE-based or EAB/ESB-based
- Instantiates easily with the MegaWizard Plug-In Manager

General Description of lpm_ram_dq

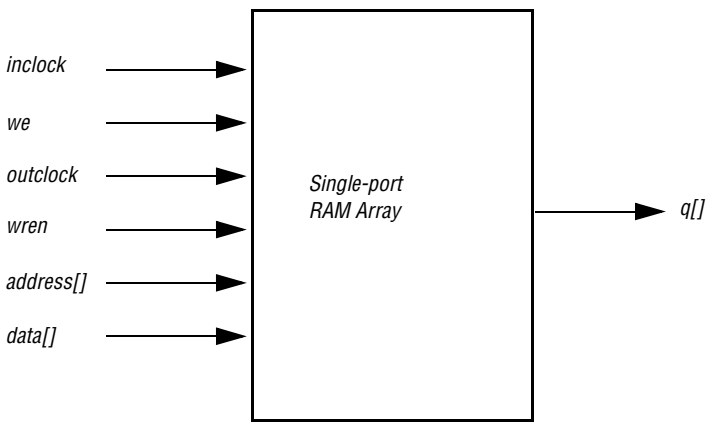
The lpm_ram_dq megafunction is a single-port RAM megafunction with separate input and output ports provided in the Quartus II software MegaWizard Plug-In Manager.

The lpm_ram_dq megafunction is used in applications that require parallel data transfer. Synchronous write operations into the memory block use the address[] and data[] ports, which are triggered by the rising edge of the inclock while the write enable (we) port is enabled. The outclock port is optional for the read operation.

For asynchronous operations, setup and hold times must be valid with respect to both edges of the write enable signal. Ideally, the data and address lines should not be changed while the (we) port is active.

When using lpm_ram_dq, you cannot access the clock enable port or clear port on the EAB registers. To access these controls, you must use the dual-port RAM, lpm_ram_dp.

Figure 1–2. lpm_ram_dq Megafunction Block Diagram



Common Applications of lpm_ram_dq

The lpm_ram_dq megafunction implements a single-port memory with separate input and output ports in the earlier devices such as APEX 20K, APEX II, ARM-based Excalibur, Mercury, ACEX 1K, FLEX 10KE, FLEX 6000, MAX 3000, and MAX 7000 devices. For other devices, Altera recommends using the altsyncram function to implement a single-port memory.

Resource Utilization & Performance of lpm_ram_dq

The lpm_ram_dq megafunction uses the following device resources:

- Embedded System Blocks (ESB) in APEX 20K, APEX II, Excalibur, and Mercury devices
- Embedded Array Blocks (EAB) in ACEX 1K and FLEX 10K devices
- DFFE primitives or latch arrays in FLEX 6000, MAX 3000, and MAX 7000 devices, or if the USE_EAB parameter is set to "OFF".

Features of lpm_ram_io

The lpm_ram_dq megafunction implements a single-port RAM function and offers many additional features, which include:

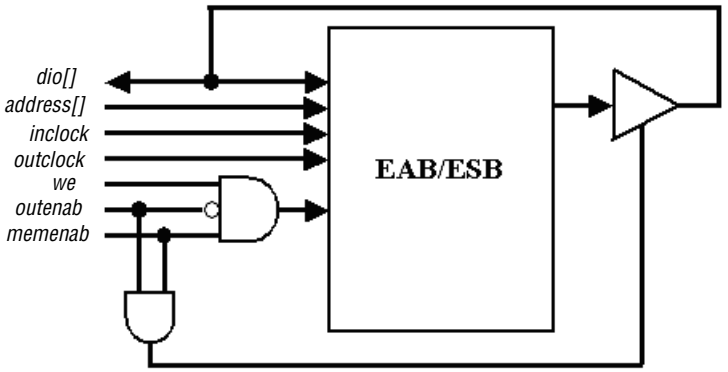
- Provides fully-parameterizable single-port RAM with single I/O port
- Provides synchronous or asynchronous operation
- Registers EAB/ESB data and/or address inputs using inclock
- Registers output data using a separate output clock
- LE-based or EAB/ESB-based
- Instantiates easily with the MegaWizard Plug-In Manager

General Description of lpm_ram_io

The lpm_ram_io megafunction is a single-port RAM megafunction with a shared input/output port provided in the Quartus II software MegaWizard Plug-In Manager.

The lpm_ram_io megafunction is used in applications where a single-port RAM with a single, shared data input/output port is required. The direction of data flow in the RAM is controlled by the outenab port. When the outenab port is low (inactive), the tri-state buffer is disabled, write operations are enabled, and the dio[] port is used as an input. When the outenab port is high (active), the tri-state buffer is enabled, read operations are enabled, and the dio[] port is used as an output that is driven by the RAM.

Figure 1–3. lpm_ram_io Megafunction Block Diagram



Common Applications of lpm_ram_io

The `lpm_ram_io` megafunction is used to implement a single-port memory with a shared input/output port in the earlier devices such as APEX 20K, APEX II, Excalibur, Mercury, ACEX 1K, FLEX 10KE, FLEX 6000, MAX 3000, and MAX 7000 devices. For other device families, Altera recommends using the `altsyncram` function to implement dual-port memory.

Resource Utilization & Performance of lpm_ram_io

The `lpm_ram_io` megafunction uses the following device resources:

- Embedded System Blocks (ESB) in APEX 20K, APEX II, Excalibur, and Mercury devices
- Embedded Array Blocks (EAB) in ACEX 1K and FLEX 10K devices
- DFFE primitives or latch arrays in FLEX 6000, MAX 3000, and MAX 7000 devices or if the `USE_EAB` parameter set to "`OFF`"

Chapter 2. Getting Started

System 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 7 or 8 operating system
- Quartus® II software version 4.1 or later

Mega Wizard Plug-In Manager Customization

You can use the MegaWizard® Plug-In Manager to set the `lpm_ram_dp`, `lpm_ram_dq`, and `lpm_ram_io` megafunction features for each RAM function 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↵`

Using the MegaWizard Plug-In Manager

This section provides descriptions for the options available in the `lpm_ram_dp`, `lpm_ram_dq`, and `lpm_ram_io` MegaWizard Plug-In Manager pages.

The lpm_ram_dp Megafunction Page Descriptions

Page 1 of the `lpm_ram_dp` MegaWizard Plug-In Manager is where you specify the device family, set the width of the data input bus and address input bus, select the clock mode, and set the read enable signal. Figure 2–1 below shows page 1 of the `lpm_ram_dp` megafunction wizard.

Figure 2–1. MegaWizard Plug-In Manager - Dual-port RAM [page 1 of 4]

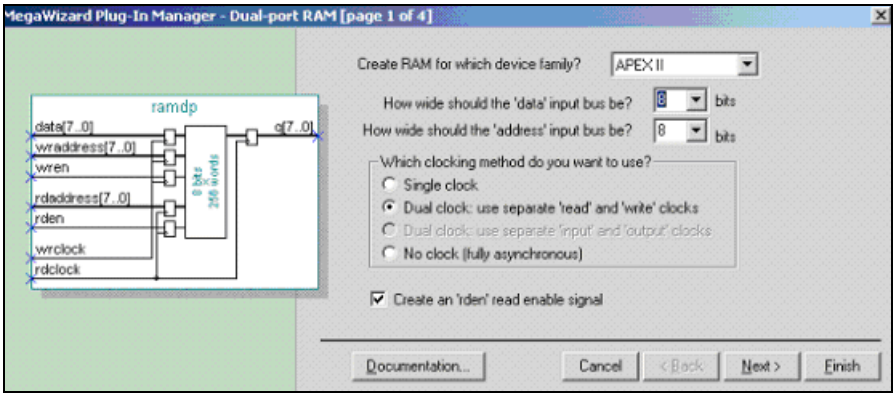


Table 2–1 below shows the features and settings of the lpm_ram_dp megafunction. Use this table, along with the hardware descriptions for the features, to determine the appropriate settings.

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

Function	Description
Create RAM for which device family?	Specify which Altera device family to use.
How wide should the 'data' input bus be?	Specify the width of data input.
How wide should the 'address' input bus be?	Specify the width of address input.
Which clocking method do you want to use?	Specify the clocking mode from single clock, dual clock, or no clock.
Create a read enable signal?	When turned on, the rden port is added.

Starting on Page 1 of the lpm_ram_dp megafunction wizard, you can generate a sample simulation waveform and launch the Quartus II Help for the lpm_ram_dp megafunction by selecting the **Generate Sample Waveforms** or **Quartus II Megafunction Reference** options from the **Documentation** button.

Page 2 of lpm_ram_dp megafunction wizard is where you specify input and output ports for registration and create a clock enable signal for wrclock and rdclock (Figure 2–2).

Figure 2–2. MegaWizard Plug-In Manager - Dual-port RAM [page 2 of 4]

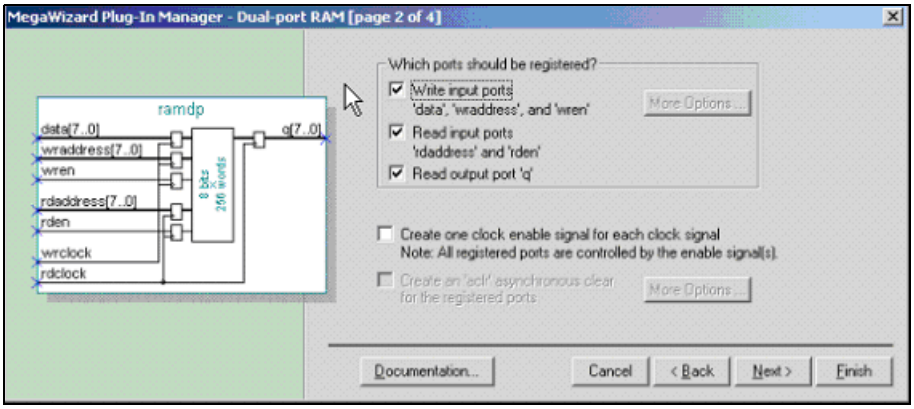


Table 2–2 below shows the features and settings for page 2 of the lpm_ram_dp megafunction wizard.

Table 2–2. lpm_ram_dp MegaWizard Plug-in Manager Page 2 Options

Function	Description
Which ports should be registered?	When turned on, the port is registered.
Create one clock enable signal for each clock signal.	When turned on, clock enable signals are created.

Page 3 of lpm_ram_dp megafunction wizard is where you specify the initial content of memory and whether this dual-port memory is implemented in logic cells or in EABs and ESBs (Figure 2–3).

Figure 2–3. MegaWizard Plug-In Manager - Dual-port RAM [page 3 of 4]

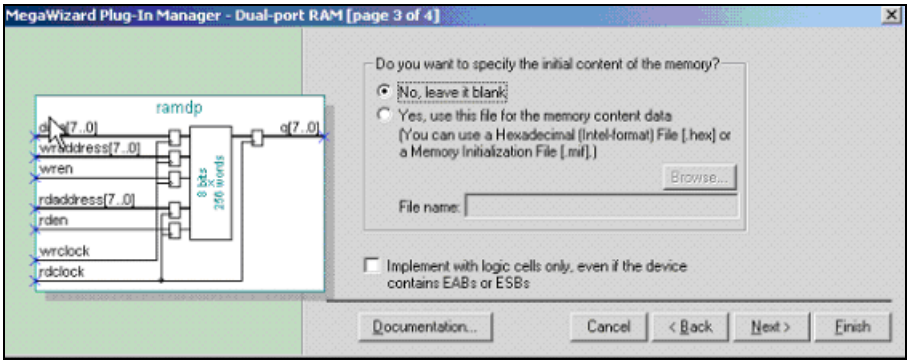


Table 2–3 below shows the features and settings for page 3 of the lpm_ram_dp megafunction wizard.

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

Function	Description
Do you want to specify the initial content of the memory?	Specify the initial content of the memory, leave it blank, or specify the Memory Initialization File (MIF)
Implement with logic cells only, even if the device contains EABs or ESBs.	When turned on, memory is implemented in logic cells. When not turned on, memory is implemented in EABs or ESBs

The lpm_ram_dq Megafunction Page Descriptions

Page 3 of the lpm_ram_dq megafunction wizard is where you specify the device family and the width of both the data input bus and the address input bus.

Figure 2–4. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 3 of 6]

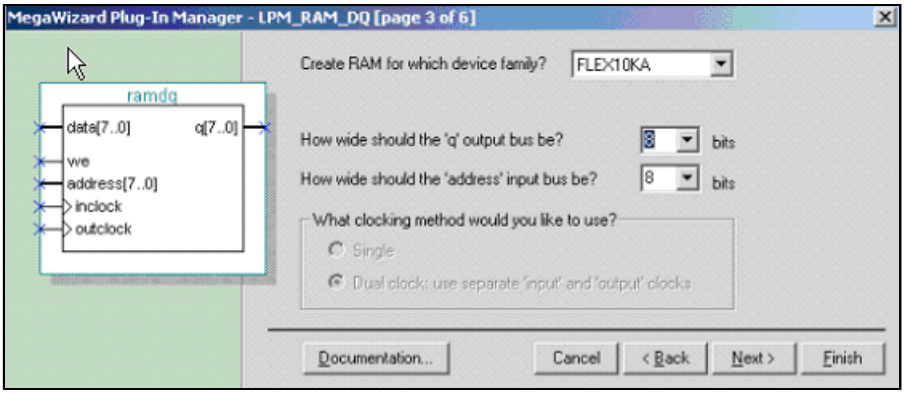


Table 2–4 below shows the features and settings for page 3 of the lpm_ram_dq megafunction wizard.

Table 2–4. lpm_ram_dq MegaWizard Plug-in Manager Page 3 Options

Function	Description
Create RAM for which device family?	Specify the Altera device family to use.
How wide should the 'q' output bus be?	Specify the width for the 'q' output bus.
How wide should the 'address' input bus be?	Specify the width for the 'address' input bus.

Page 4 of lpm_ram_dq megafunction wizard is where you specify the ports for registration.

Figure 2–5. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 4 of 6]

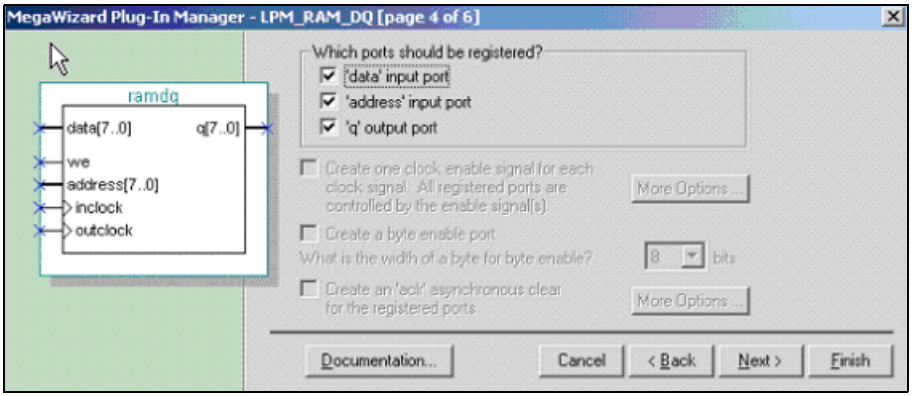


Table 2–5 below shows the features and settings for page 4 of the `lpm_ram_dq` megafunction wizard.

Table 2–5. lpm_ram_dq MegaWizard Plug-in Manager Page 4 Options

Function	Description
Which ports should be registered?	When the port is turned on, it is registered.

Page 5 of `lpm_ram_dq` megafunction wizard is where you specify the initial content of memory and whether this single-port memory is implemented in logic cells or in EABs and ESBs.

Figure 2–6. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 5 of 6]

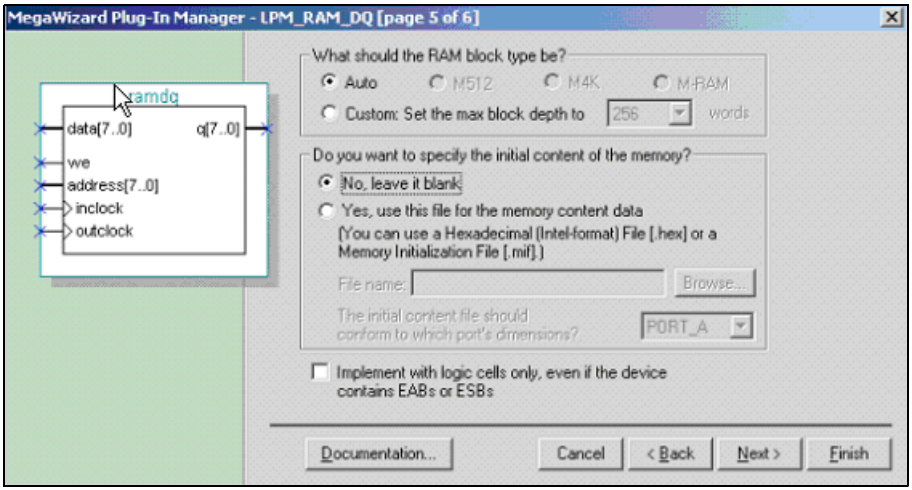


Table 2–6 below shows the features and settings for page 5 of the `lpm_ram_dq` megafunction wizard.

Table 2–6. lpm_ram_dq MegaWizard Plug-in Manager Page 5 Options

Function	Description
What should the RAM block type be?	'auto': lets the Quartus II program select the RAM block type and 'custom': lets you specify the maximum block depth.
Do you want to specify the initial content of the memory?	Specify the memory initialization file, or leave the memory blank.
Implement with logic cells only if the device contains EABs or ESBs	When this option is turned on, the memory is implemented using logic cells. When this option is turned off, the memory is implemented using EABs or ESBs.

The lpm_ram_io Megafunction Page Descriptions

Page 3 of the `lpm_ram_io` megafunction wizard is where you specify the device family and the width of both the data input bus and the address input bus.

Figure 2–7. MegaWizard Plug-In Manager - LPM_RAM_IO [page 3 of 6]

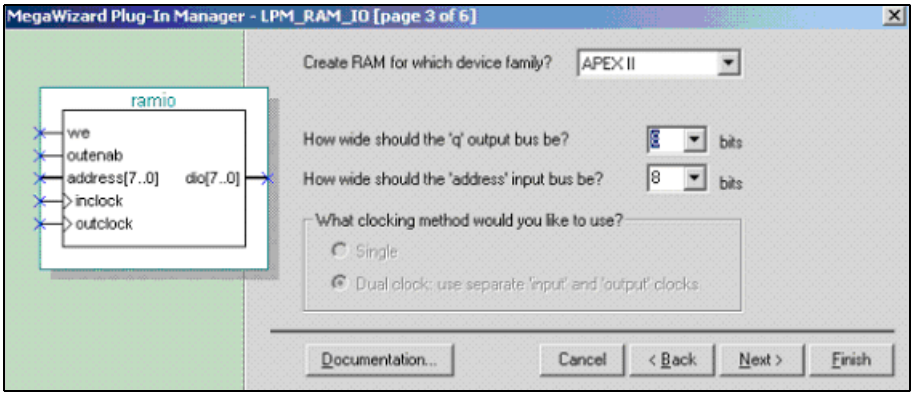


Table 2–7 below shows the features and settings for page 3 of the lpm_ram_io megafunction wizard.

Table 2–7. lpm_ram_io MegaWizard Plug-in Manager Page 3 Options

Function	Description
Create RAM for which device family?	Specify the Altera device family to use.
How wide should the 'q' output bus be?	Specify the width for the 'q' output bus.
How wide should the 'address' input bus be?	Specify the width for the 'address' input bus.

Page 4 of lpm_ram_io megafunction wizard is where you specify the ports for registration.

Figure 2–8. MegaWizard Plug-In Manager - LPM_RAM_IO [page 4 of 6]

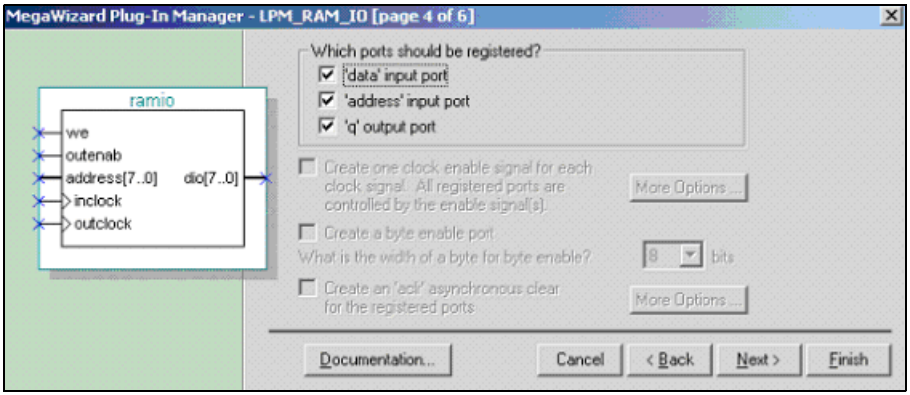


Table 2–8 below shows the features and settings for page 4 of the lpm_ram_io megafunction wizard.

Table 2–8. lpm_ram_io MegaWizard Plug-in Manager Page 4 Options

Function	Description
Which ports should be registered?	When this option is turned on, it is registered.

Page 5 of lpm_ram_io megafunction wizard is where you specify the initial content of memory and specify where to map the memory is logic cells, EABs, or ESBs.

Figure 2–9. MegaWizard Plug-In Manager - LPM_RAM_IO [page 5 of 6]

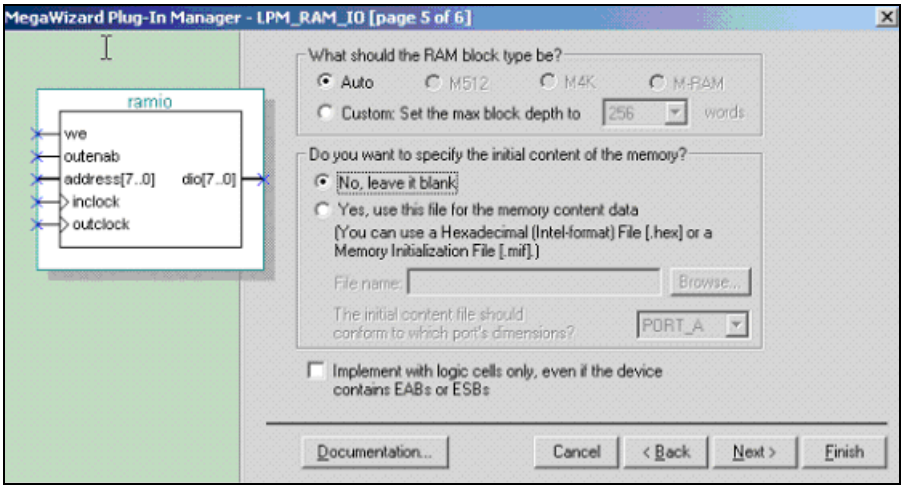


Table 2–9 below shows the features and settings for page 5 of the `lpm_ram_io` megafunction wizard.

Table 2–9. lpm_ram_io MegaWizard Plug-in Manager Page 5 Options

Function	Description
What should the RAM block type be?	'auto': lets the Quartus II program select the RAM block type and 'custom': lets you specify the maximum block depth.
Do you want to specify the initial content of the memory?	Specify the memory initialization file, or leave the memory blank.
Implement with logic cells only if the device contains EABs or ESBs.	When turned on, the memory is implemented using logic cells. When turned off, the memory is implemented using EABs or ESBs.

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. 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—that generally provide improved performance compared with basic logic elements.

Refer to 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 3rd party synthesis support chapters in the Synthesis section in Volume 1 of the *Quartus II Handbook*.

Instantiating Megafunctions Using the Port & Parameter Definition for lpm_ram_dp

VHDL Component Declaration for the `lpm_ram_dp` megafunction

```
COMPONENT lpm_ram_dp
   GENERIC ( LPM_WIDTH: POSITIVE;
      LPM_WIDTHAD: POSITIVE;
      LPM_NUMWORDS: NATURAL := 0;
      LPM_INDATA: STRING := "REGISTERED";
      LPM_OUTDATA: STRING := "REGISTERED";
      LPM_RDADDRESS_CONTROL: STRING := "REGISTERED";
      LPM_WRADDRESS_CONTROL: STRING := "REGISTERED";
      LPM_FILE: STRING := "UNUSED";
      LPM_TYPE: STRING := "LPM_RAM_DP";
      LPM_HINT: STRING := "UNUSED");
   PORT ( data: IN STD_LOGIC_VECTOR(LPM_WIDTH-1 DOWNTO 0);
      rdaddress, wraddress: IN STD_LOGIC_VECTOR(LPM_WIDTHAD-1 DOWNTO 0);
      rdclock, wrclock: IN STD_LOGIC := '0';
      rden, rdclken, wrclken: IN STD_LOGIC := '1';
      wren: IN STD_LOGIC;
      q: OUT STD_LOGIC_VECTOR(LPM_WIDTH-1 DOWNTO 0));
```

```
END COMPONENT;
```

Verilog Module Declaration for the lpm_ram_dp megafunction

```
module lpm_ram_dp (wren, data, wraddress, wrclock, wrclken, rden,
 rdaddress, rdclock, rdclken, q);

input [LPM_WIDTH-1:0] data;
input [LPM_WIDTHAD-1:0] rdaddress, wraddress;
input wren, wrclock, wrclken, rden;
input rdclock, rdclken;
output [LPM_WIDTH-1:0] q;

endmodule
```

Instantiating Megafunctions Using the Port & Parameter Definition for lpm_ram_dq

VHDL Component Declaration for the lpm_ram_dq megafunction

```
COMPONENT lpm_ram_dq
   GENERIC (LPM_WIDTH: POSITIVE;
      LPM_WIDTHAD: POSITIVE;
      LPM_NUMWORDS: NATURAL := 0;
      LPM_INDATA: STRING := "REGISTERED";
      LPM_ADDRESS_CONTROL: STRING := "REGISTERED";
      LPM_OUTDATA: STRING := "REGISTERED";
      LPM_FILE: STRING := "UNUSED";
      LPM_TYPE: STRING := "LPM_RAM_DQ";
      LPM_HINT: STRING := "UNUSED");
    PORT (data: IN STD_LOGIC_VECTOR(LPM_WIDTH-1 DOWNTO 0);
      address: IN STD_LOGIC_VECTOR(LPM_WIDTHAD-1 DOWNTO 0);
      inclock, outclock: IN STD_LOGIC := '0';
      we: IN STD_LOGIC;
      q: OUT STD_LOGIC_VECTOR(LPM_WIDTH-1 DOWNTO 0));
END COMPONENT;
```

Verilog Module Declaration for the lpm_ram_dq megafunction

```
  module lpm_ram_dq (data, address, inclock, outclock, we, q);
     input data[LPM_WIDTH-1:0];
         input address[LPM_WIDTHAD-1:0];
         input inclock;
         input outclock;
         input we;
         output q[LPM_WIDTH-1:0];
   endmodule
```

Instantiating Megafunctions Using the Port & Parameter Definition for lpm_ram_io

VHDL Component Declaration for the lpm_ram_io megafunction

```
COMPONENT lpm_ram_io
   GENERIC (LPM_WIDTH: POSITIVE;
      LPM_WIDTHAD: POSITIVE;
      LPM_NUMWORDS: NATURAL := 0;
      LPM_INDATA: STRING := "REGISTERED";
      LPM_ADDRESS_CONTROL: STRING := "REGISTERED";
      LPM_OUTDATA: STRING := "REGISTERED";
      LPM_FILE: STRING; := "UNUSED"
      LPM_TYPE: STRING := "LPM_RAM_IO";
      LPM_HINT: STRING := "UNUSED");
   PORT (address: IN STD_LOGIC_VECTOR(LPM_WIDTHAD-1 DOWNTO 0);
      inclock, outclock: IN STD_LOGIC := '0';
      memenab, outenab : IN STD_LOGIC := '1';
      we: IN STD_LOGIC;
      dio: INOUT STD_LOGIC_VECTOR(LPM_WIDTH-1 DOWNTO 0));
END COMPONENT;
```

Verilog Module Declaration for the `lpm_ram_io` megafunction

```
module lpm_ram_io (address, we, inclock, outclock, outenab,
           memenab, dio);

input [LPM_WIDTHAD-1:0] address;
input we, inclock, outclock, outenab, memenab;
inout [LPM_WIDTH-1:0] dio;

endmodule
```

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 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 "Integrated Synthesis" chapter of the *Quartus II Handbook*.

EDA Simulation

Depending on the 3rd 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 on the files that are needed and the directories where those files are located.

SignalTap II Embedded Logic Analyzer

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

To monitor signals from your Altera megafunctions, 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 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 of the *Quartus II Handbook*.

In-System Updating of Memory and Constants

FPGA designs are growing larger in density and are becoming more complex. Designers and verification engineers require more access to the design that is programmed in the device to quickly identify, test, and resolve issues. The In-System Updating of Memory and Constants capability of the Quartus II software provides you with a non-intrusive method of accessing your RAM within the Altera FPGA. With the In-System Memory Content Editor, you can capture, analyze, and update RAM data while your system is running at full speed.

To gain access to your RAM megafunction, enable the In-System Updating of Memory and Constants feature within the MegaWizard Plug-in Manager. The Quartus II software will then modify your RAM (in the background) so you have access to it while the FPGA is processing. You can read, write, or update the contents of your RAM multiple times without having to reconfigure your FPGA.

For more information on viewing and modifying internal memories and constants, refer to the "In-System Updating of Memory and Constants" chapter of the *Quartus II Handbook*.

Design Examples for the RAM Megafunctions

This section presents a design example that uses the `lpm_ram_dp` megafunction to generate a dual-port memory. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described in detail. When you are finished with this example, you can incorporate it into 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 at the following URL: www.altera.com.

Example for lpm_ram_dp Dual-port Memory

The objective of the examples is to instantiate the `lpm_ram_dp` megafunction using the MegaWizard Plug-In Manger. The `lpm_ram_dp` examples illustrate: single clock with unregistered output mode, single clock with registered output mode, separate clock with unregistered output mode, separate clock with registered output mode, and asynchronous mode.

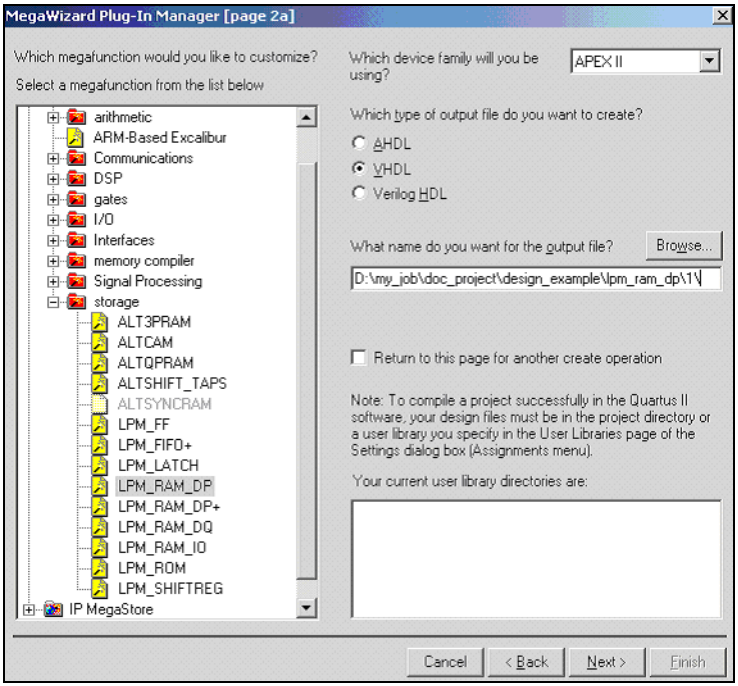
In this example, you perform the following activities:

- Generate a dual-port memory using the `lpm_ram_dp` megafunction and the Megawizard Plug-In Manager
- Implement the dual-port memory by assigning the APEX II device to the project and compiling the project
- Simulate the dual-port memory design

Generate the lpm_ram_dp Dual-port Memory

1. Open the project file **lpm_ram_dp\1\top.qpf**.
2. Open the top-level file **top.bdf**. This is an incomplete file that you will complete as a part of this example.
3. Double-click on a blank area in the block design file (**.bdf**).
4. Choose **MegaWizard Plug-In Manager** in the **Symbol** window.
5. In the window that appears, turn on **Create a new custom megafunction variation** in the **What action do you want to perform?** section.
6. Click **Next**.
7. On page 2a, expand the Arithmetic folder and select the `LPM_RAM_DP` megafunction (Figure 2–10).
8. Select APEX II in the **Which device family will you be using?** list.
9. Turn on **VHDL** option under the **What type of output file do you want to create?** section.
10. Set the output file name to ***<project directory>*\simple_clk**.
11. Click **Next**.

Figure 2–10. MegaWizard Plug In Manager [page 2a]

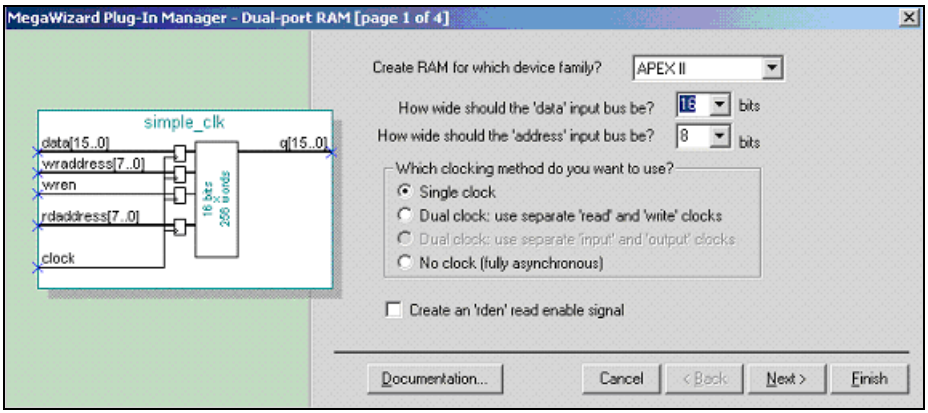


12. On page 1, select 16 bits for **How wide should the 'data' input bus be?** section (Figure 2–11).

13. Select 8 bits for **How wide should the 'address' input bus be?** section.

14. Click **Next**.

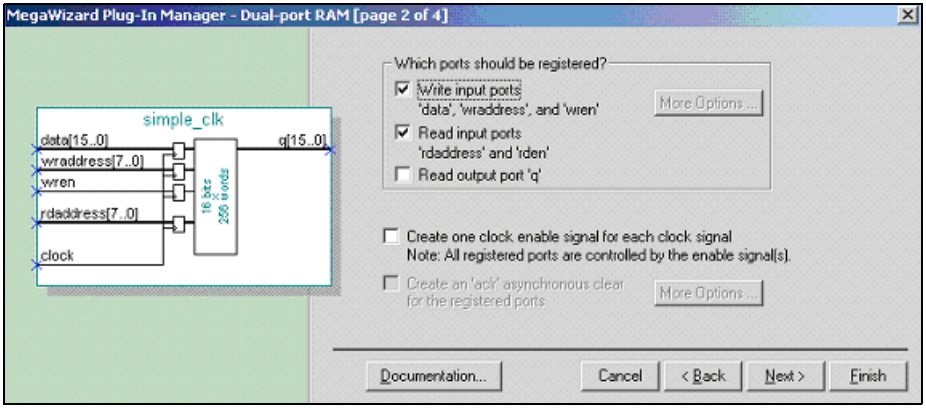
Figure 2–11. MegaWizard Plug-In Manager - Dual-port RAM [page 1 of 4]



15. On Page 2, turn on the **Write input ports** and **Read input ports** options under the **Which ports should be registered?** section (Figure 2–12).

16. Click **Next**.

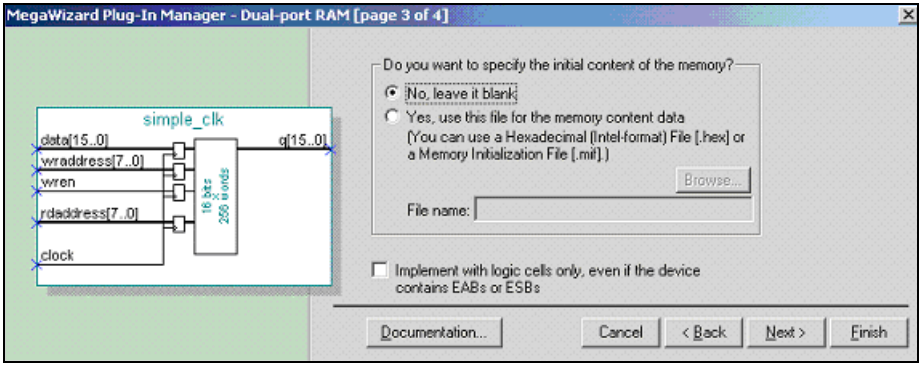
Figure 2–12. MegaWizard Plug-In Manager - Dual-port RAM [page 2 of 4]



17. On page 3, turn on the **No, leave it blank** option in the **Do you want to specify the initial content of the memory?** section (Figure 2–13).

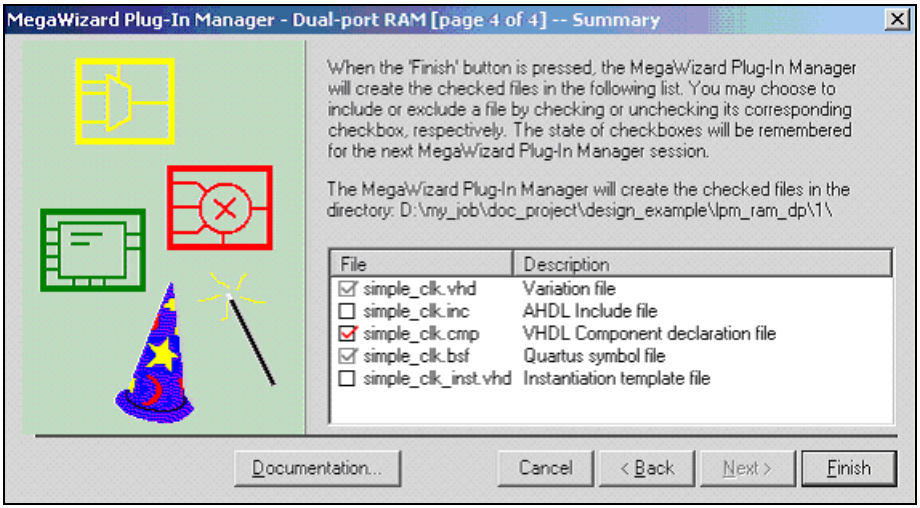
18. Click **Next**.

Figure 2–13. MegaWizard Plug-In Manager - Dual-port RAM [page 3 of 4]



19. On page 4, turn on the options corresponding to a **VHDL** design file(**.vhd**), Quartus symbol file, and VHDL component file, so they can be created for the project (Figure 2–14).

Figure 2–14. MegaWizard Plug-In Manager [page 4 of 4] Summary



20. Click **Finish**. The `lpm_ram_dp` megafunction is built.

21. Move the mouse to place the RAM symbol in between the input and output ports of the **top.bdf** file. Click the left mouse button to place the RAM symbol. You have now completed the design file.

19. Select **Save** (**File** menu) to save the design.

Implement the lpm_ram_dp Dual-port Memory

Next, assign the EP2A15B724C7 device to the project and compile the project.

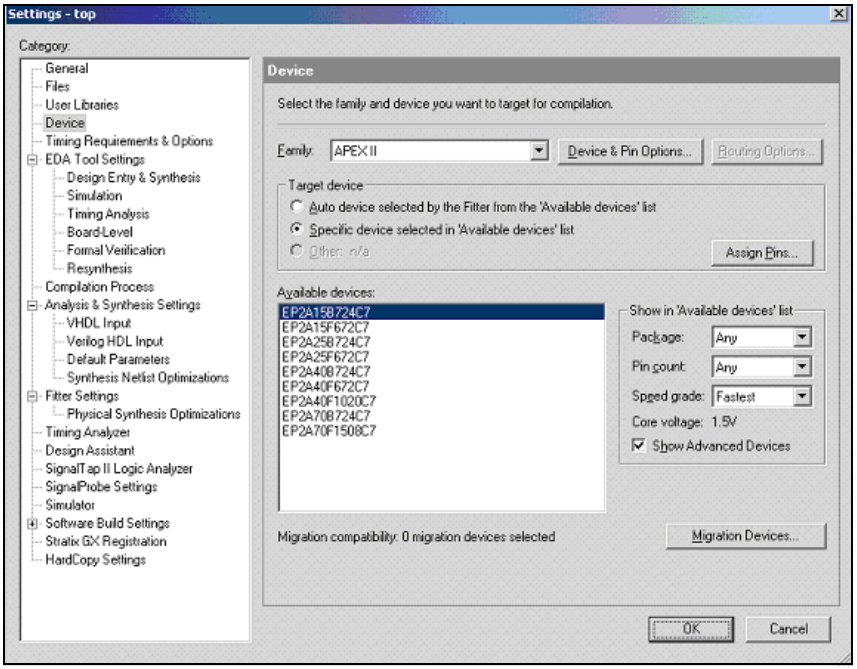
1. Select **Settings** (**Assignments** menu) to open the **Settings** dialog box.

2. Click the **Devices** category. Ensure that APEX II is selected in the Family field (Figure 2–15).

3. Select **EP2A15B724C7** under the '**Available devices**' list in the **Target device** section.

4. Click **OK**.

Figure 2–15. Settings - top

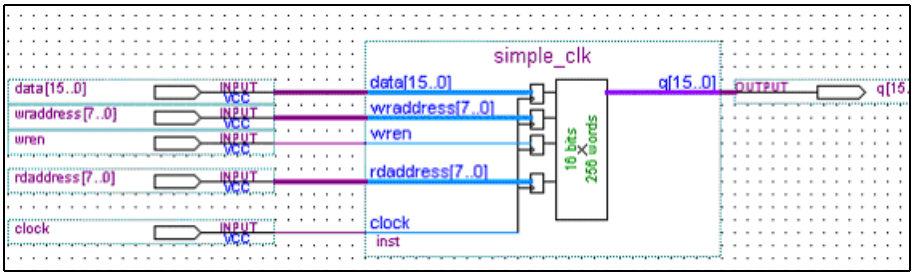


5. Select **Start Compilation** (**Processing** menu) to compile the design.

6. Click **OK** when the **Full compilation was successful** message box appears.

7. In the **Compilation Report** window, you can expand the Fitter section and click on the Floorplan View option to view how the module is implemented in the APEX II device.

Figure 2–16. Top-level Design for Single Clock Mode (Unregistered Output)

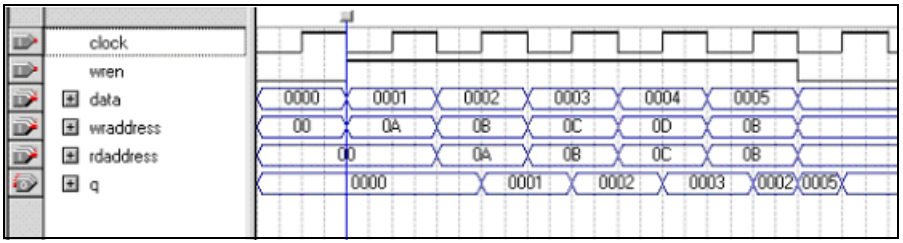


Functional Results - Simulate the lpm_ram_dp Dual-port Memory

Finally, simulate the design to verify the results. Setup the Quartus II Simulator by performing the following steps:

1. From the Processing menu, select **Generate Functional Simulation Netlist**.
2. Click **OK**.
3. Select **Wizards > Simulator Setting Wizard** (**Assignments** menu).
4. In the Category section, highlight the **Simulator** category.
5. Select **Functional** under the **Simulation mode**.
6. Select **Start Simulation** (**Processing** menu) to start the simulation.
7. Click **OK** when the **Simulation was successful** message box appears.
8. In the **Simulation Report** window, look at the simulation output waveform and verify the results. Figure 2–17 shows the expected simulation results.

Figure 2–17. Functional Waveform for Single Clock Mode (Unregistered Output)



When the dual port memory is configured in other modes, the steps to run the project are similar. For the additional modes, just the top-level schematic and functional simulation waveforms are illustrated below.

Figure 2–18. Top-level Design for Single Clock Mode (Registered Output)

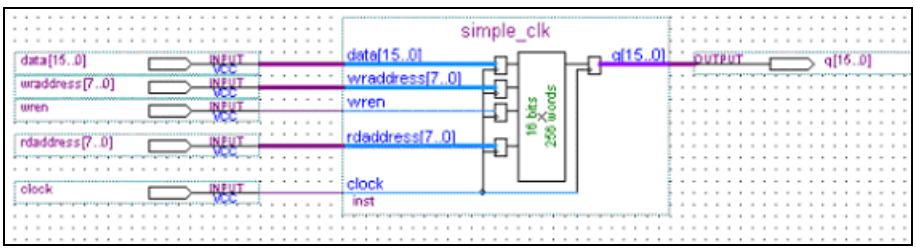


Figure 2–19. Functional Waveform for Single Clock Mode (Registered Output)

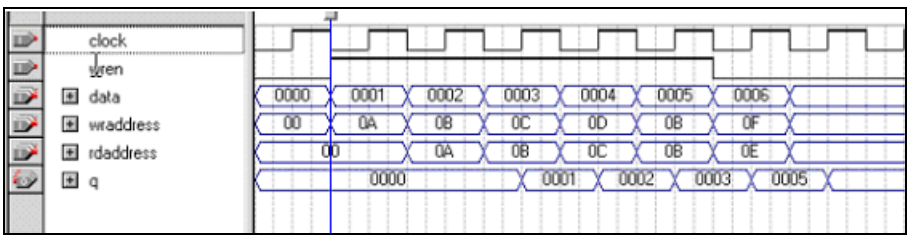


Figure 2–20. Top-level Design for Separate Clock Mode (Registered Output)

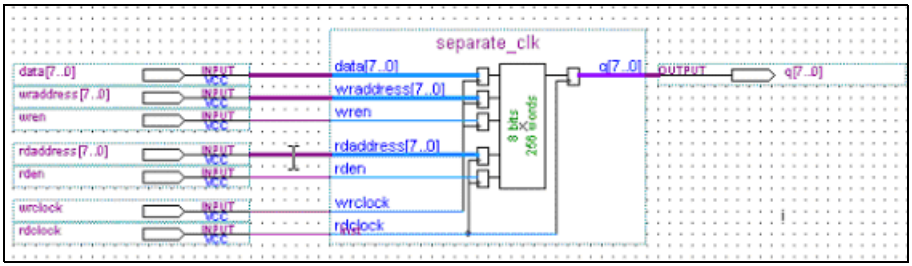


Figure 2–21. Functional Waveform for Separate Clock Mode (Registered Output)

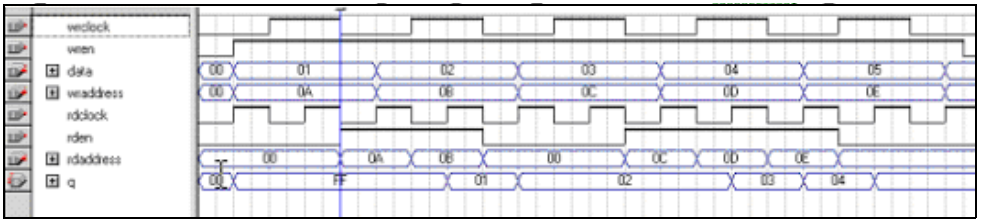


Figure 2–22. Top-level Design for Separate Clock Mode (Unregistered Output)

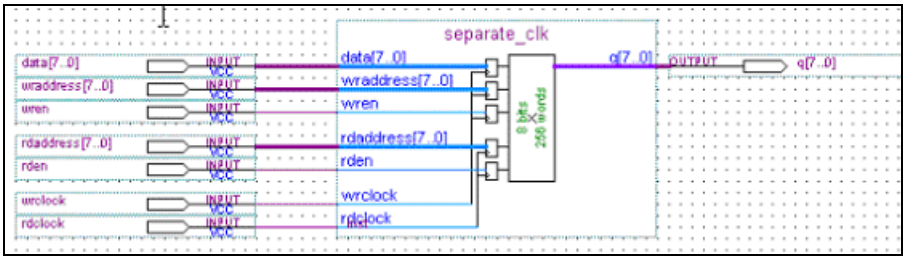


Figure 2–23. Functional Waveform for Separate Clock Mode (Unregistered Output)

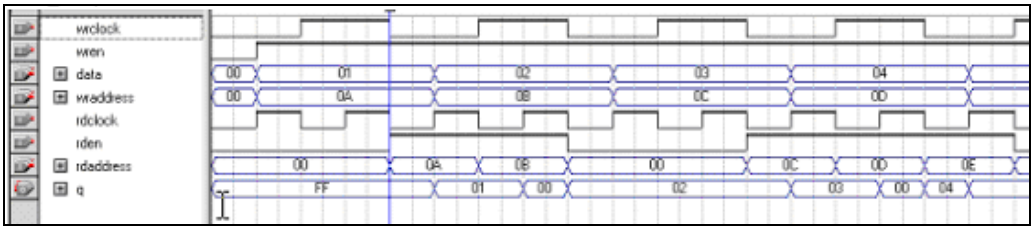


Figure 2–24. Top-level Design for Asynchronous Clock Mode

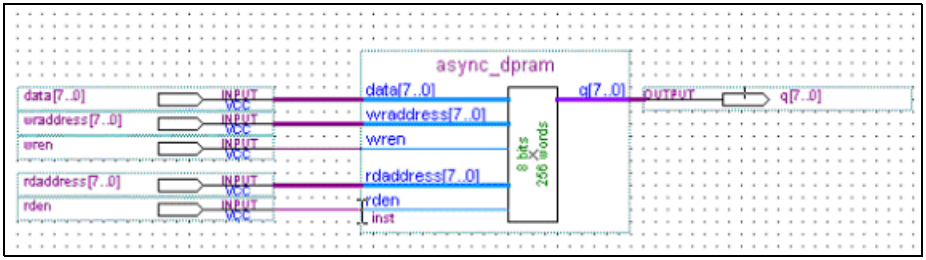
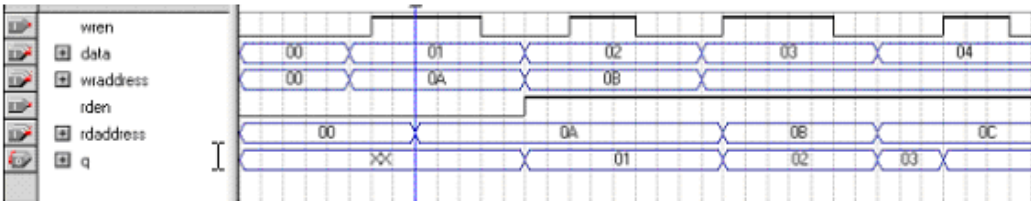


Figure 2–25. Functional Waveform for Asynchronous Clock Mode



Example for lpm_ram_dq Single-port Memory

The objective of this examples is to generates a single-port memory with separate input and output port using the `lpm_ram_dq` megafunction in the MegaWizard Plug-In Manger. The `lpm_ram_dq` examples illustrate unregistered output mode, registered output mode, and asynchronous mode.

In this example, you perform the following activities:

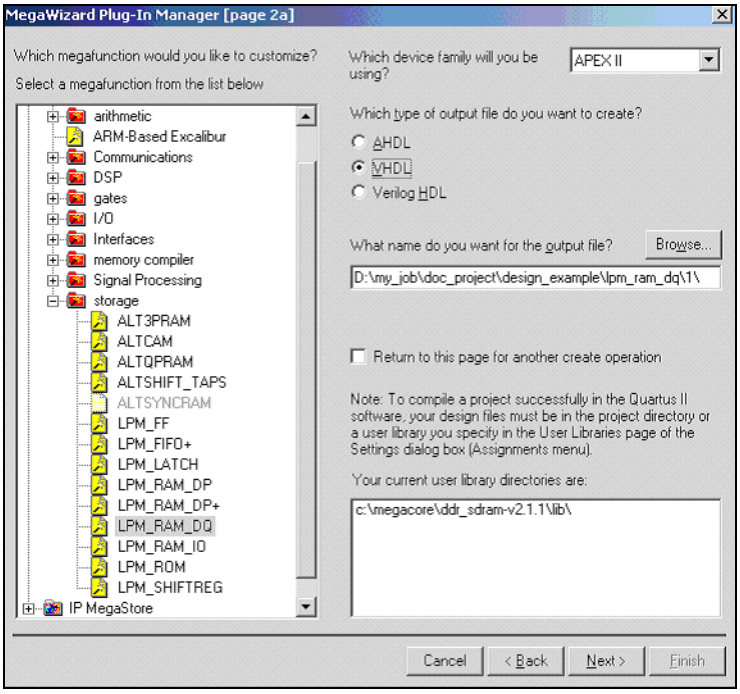
- Generate a single-port memory using the `lpm_ram_dp` megafunction and the Megawizard Plug-In Manager

- Implement the single-port memory by assigning the APEX II device to the project and compiling the project
- Simulate the single-port memory design

Generate the lpm_ram_dq Single-port Memory

1. Open the project file **lpm_ram_dq\1\top.qpf**.
2. Open the top-level file **top.bdf**. This is an incomplete file that you will complete as a part of this example.
3. Double-click on a blank area in the block design file (**.bdf**).
4. Choose **MegaWizard Plug-In Manager** in the **Symbol** window.
5. In the window that appears, turn on **Create a new custom megafunction variation** in the **What action do you want to perform?** section.
6. Click **Next**.
7. On page 2a, expand the Arithmetic folder and select the `LPM_RAM_DQ` megafunction (Figure 2–26).
8. Select **APEX II** in the **Which device family will you be using?** list.
9. Turn on **VHDL** option under the **What type of output file do you want to create?** section.
10. Set the output file name to **<*project directory*>\my_lpm_ram_dq**.
11. Click **Next**.

Figure 2–26. MegaWizard Plug In Manager [page 2a]

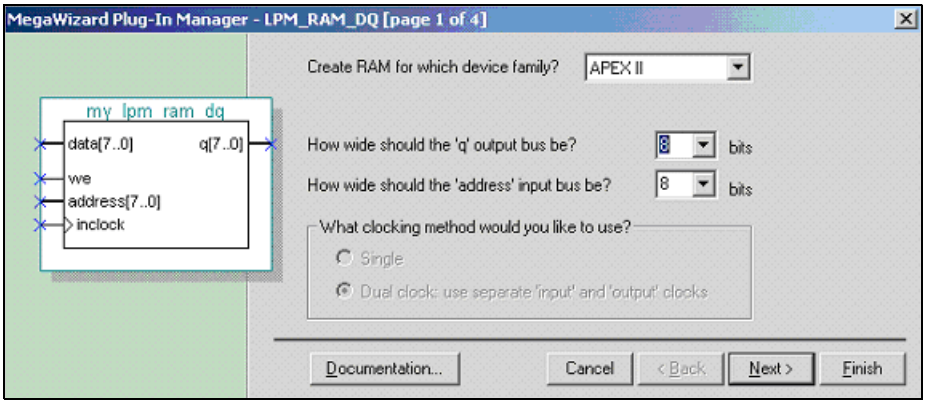


12. On page 1, select 8 bits for the **How wide should the 'q' output bus be?** option (Figure 2–27).

13. Select 8 bits for the **How wide should the 'address' input bus be?** option.

14. Click **Next**.

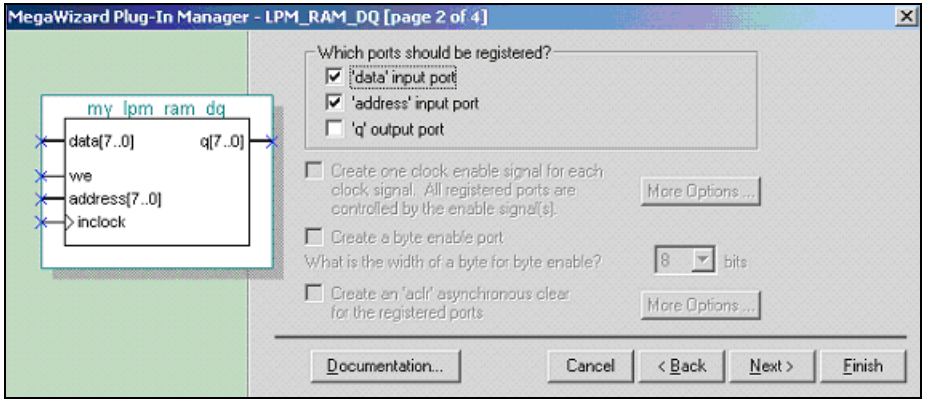
Figure 2–27. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 1 of 4]



15. On Page 2, turn on **'data' input port** and **'address' input port** under the **Which ports should be registered?** section (Figure 2–28).

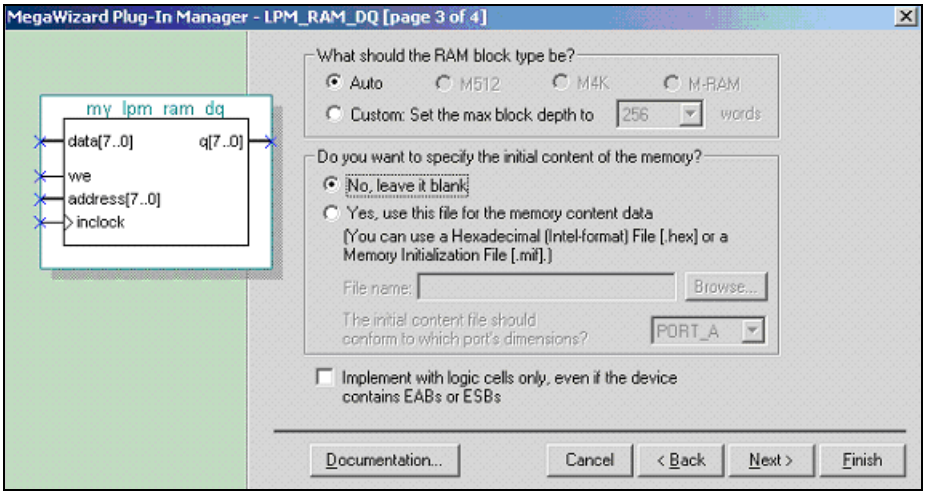
16. Click **Next**.

Figure 2–28. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 2 of 4]



17. On page 3, turn on the **No, leave it blank** option in the **Do you want to specify the initial content of the memory?** section (Figure 2–29).
18. Turn on **Auto** for the **What should the ram block type be?** section.
19. Click **Next**.

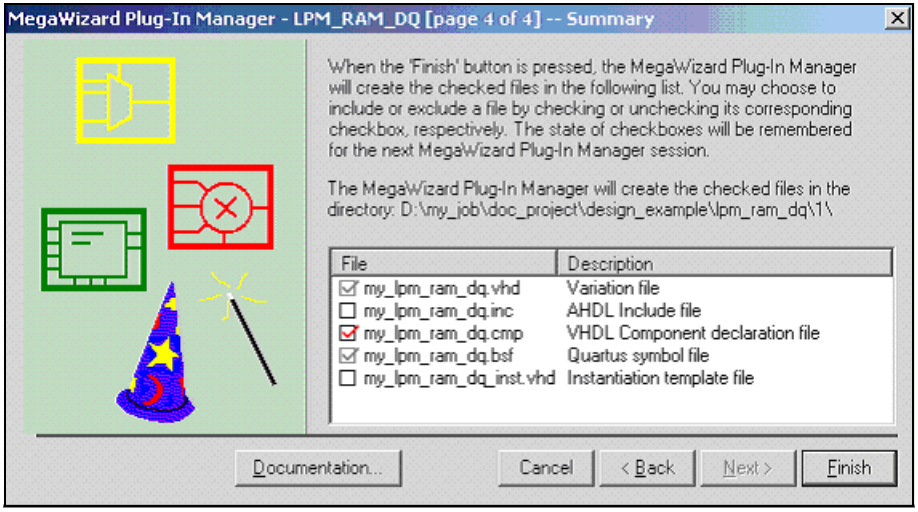
Figure 2–29. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 3 of 4]



20. On page 4, turn on the options corresponding to a **VHDL** design file(**.vhd**), Quartus symbol file, and VHDL component file, so they can be created for the project (Figure 2–30).

21. Click Next.

Figure 2–30. MegaWizard Plug-In Manager - LPM_RAM_DQ [page 4 of 4]



22. Click **Finish**. The `lpm_ram_dq` megafunction is built.

23. Move the mouse to place the RAM symbol in between the input and output ports of the **top.bdf** file. Click the left mouse button to place the RAM symbol. You have now completed the design file.

24. Select **Save** (**File** menu) to save the design.

Implement the lpm_ram_dq Single-port Memory

Next, assign the EP2A15B724C7 device to the project and compile the project.

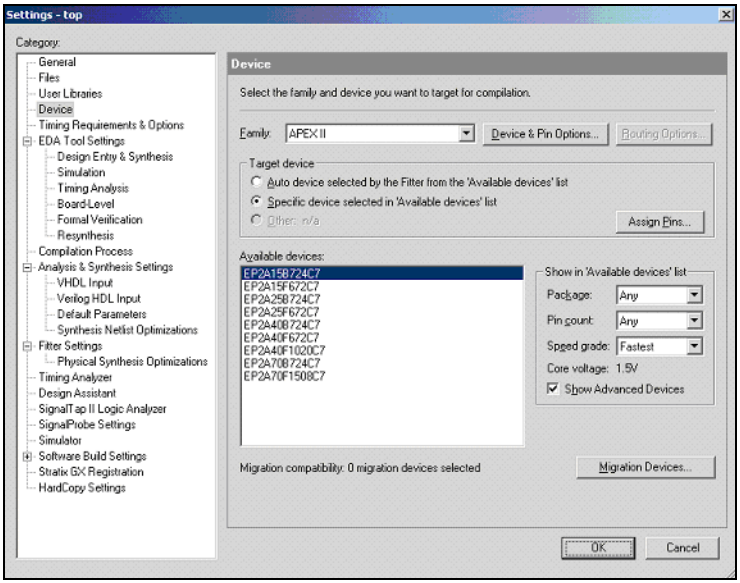
1. Select **Settings** (**Assignments** menu) to open the **Settings** dialog box.

2. Click the **Devices** category. Ensure that APEX II is selected in the Family field (Figure 2–31).

3. Select **EP2A15B724C7** under the '**Available devices**' list in the **Target device** section.

4. Click **OK**.

Figure 2–31. Settings - top

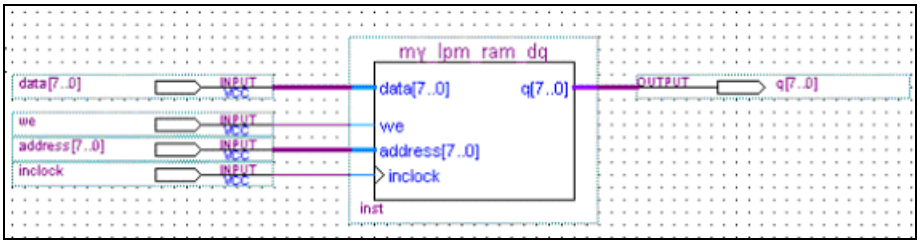


5. Select **Start Compilation** (**Processing** menu) to compile the design.

6. Click **OK** when the **Full compilation was successful** message box appears.

7. In the **Compilation Report** window, you can expand the Fitter section and click on the Floorplan View option to view how the module is implemented in the APEX II device.

Figure 2–32. Top-level Design for Read/Write Cycle Timing (Unregistered Output)

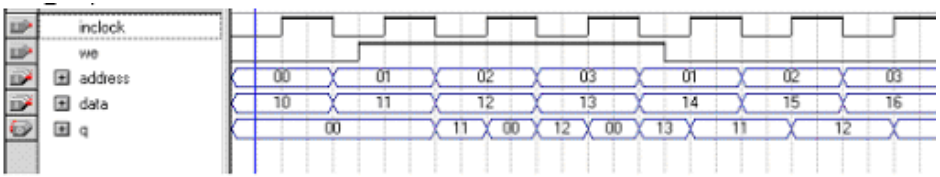


Functional Results - Simulate the lpm_ram_dq Single-port Memory

Finally, simulate the design to verify the results. Setup the Quartus II Simulator by performing the following steps:

1. From the Processing menu, select **Generate Functional Simulation Netlist**.
2. Click **OK**.
3. Select **Wizards > Simulator Setting Wizard** (**Assignments** menu).
4. In the Category section, highlight the **Simulator** category.
5. Select **Functional** under the **Simulation mode**.
6. Select **Start Simulation** (**Processing** menu) to start the simulation.
7. Click **OK** when the **Simulation was successful** message box appears.
8. In the **Simulation Report** window, look at the simulation output waveform and verify the results. Figure 2–33 shows the expected simulation results.

Figure 2–33. Functional Waveform for Read/Write Cycle Timing (Unregistered Output)



When the single-port memory is configured in other modes, the steps to run the project are similar. For the additional modes, just the top-level schematic and functional simulation waveforms are illustrated below.

Figure 2–34. Top-level Design for Synchronous Read/Write Cycle (Registered Output)

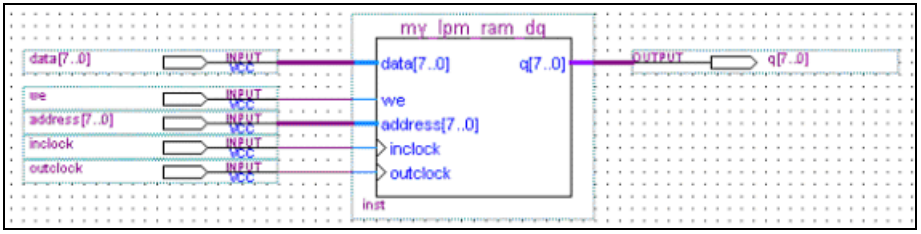


Figure 2–35. Functional Waveform for Synchronous Read/Write Cycle Timing (Registered Output)

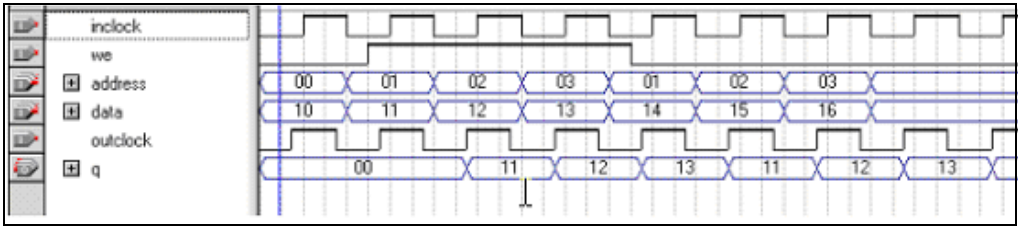


Figure 2–36. Top-level Design for Asynchronous Read/Write Cycle Timing

Figure 2–37. Functional Waveform for Asynchronous Read/Write Cycle Timing

Example for lpm_ram_io Single-port Memory

The objective of this examples is to generates a single-port memory with shared input and output port using the `lpm_ram_io` megafunction in the MegaWizard Plug-In Manger. The `lpm_ram_io` examples illustrate synchronous mode and asynchronous mode.

In this example, you perform the following activities:

- Generate a single-port memory using the `lpm_ram_io` megafunction and the Megawizard Plug-In Manager
- Implement the single-port memory by assigning the APEX II device to the project and compiling the project
- Simulate the single-port memory design

Generate the lpm_ram_io Single-port Memory

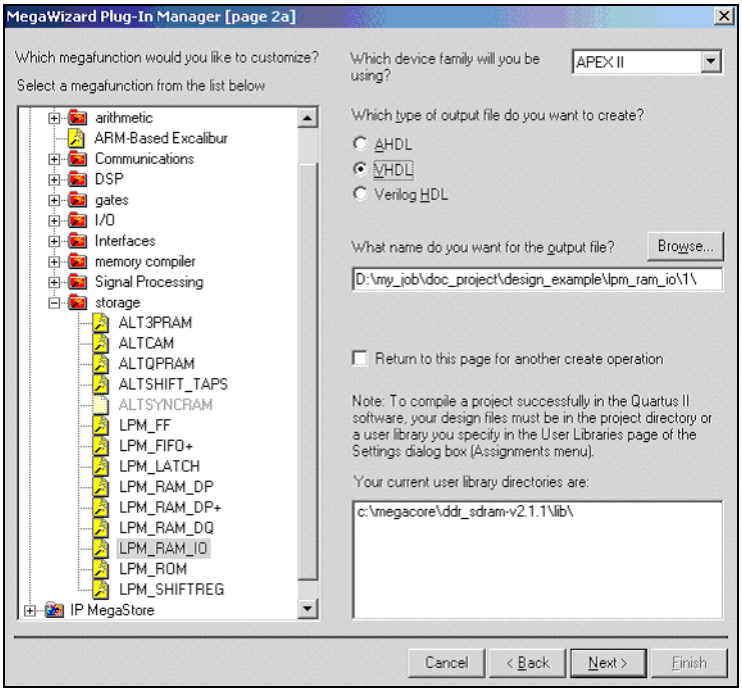
1. Open the project file **lpm_ram_io\1\top.qpf**.
2. Open the top-level file **top.bdf**. This is an incomplete file that you will complete as a part of this example.
3. Double-click on a blank area in the block design file (**.bdf**).
4. Choose **MegaWizard Plug-In Manager** in the **Symbol** window.
5. In the window that appears, turn on **Create a new custom megafunction variation** in the **What action do you want to perform?** section.
6. Click **Next**.
7. On page 2a, expand the Arithmetic folder and select the `LPM_RAM_IO` megafunction (Figure 2–38).
8. Select **APEX II** in the **Which device family will you be using?** list.

9. Turn on **VHDL** option under the **What type of output file do you want to create?** section.

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

11. Click **Next**.

Figure 2–38. MegaWizard Plug In Manager [page 2a]

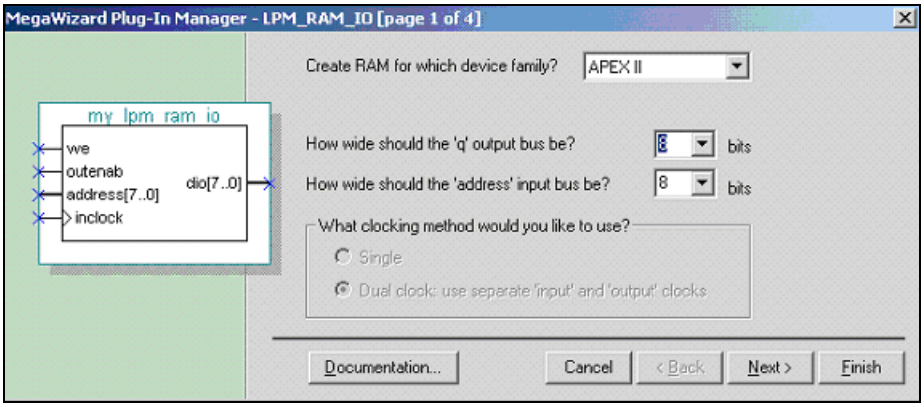


12. On page 1, select **8 bits** for the **How wide should the 'q' output bus be?** option (Figure 2–39).

13. Select **8 bits** for the **How wide should the 'address' input bus be?** option.

14. Click **Next**.

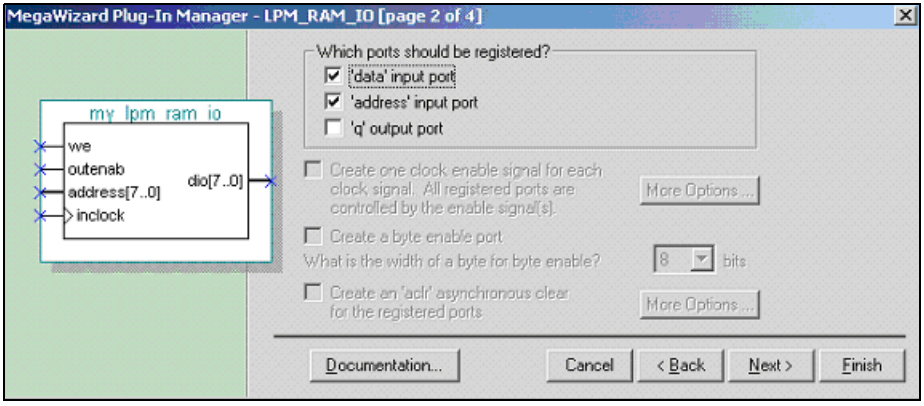
Figure 2–39. MegaWizard Plug-In Manager - LPM_RAM_IO [page 1 of 4]



15. On Page 2, turn on the **'data' input port** and **'address' input port** under the **Which ports should be registered?** section (Figure 2–40).

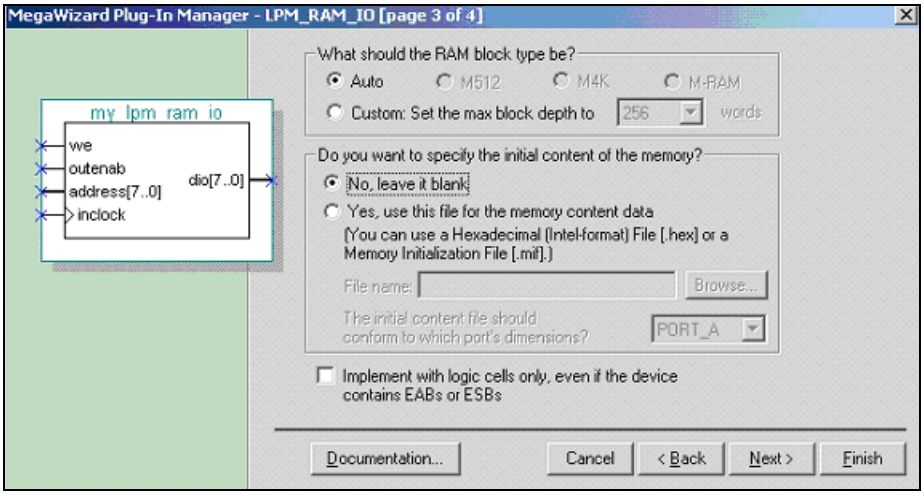
16. Click **Next**.

Figure 2–40. MegaWizard Plug-In Manager - LPM_RAM_IO [page 2 of 4]



17. On page 3, turn on the **No, leave it blank** option in the **Do you want to specify the initial content of the memory?** section (Figure 2–41).
18. Turn on **Auto** for the **What should the ram block type be?** section.
19. Click **Next**.

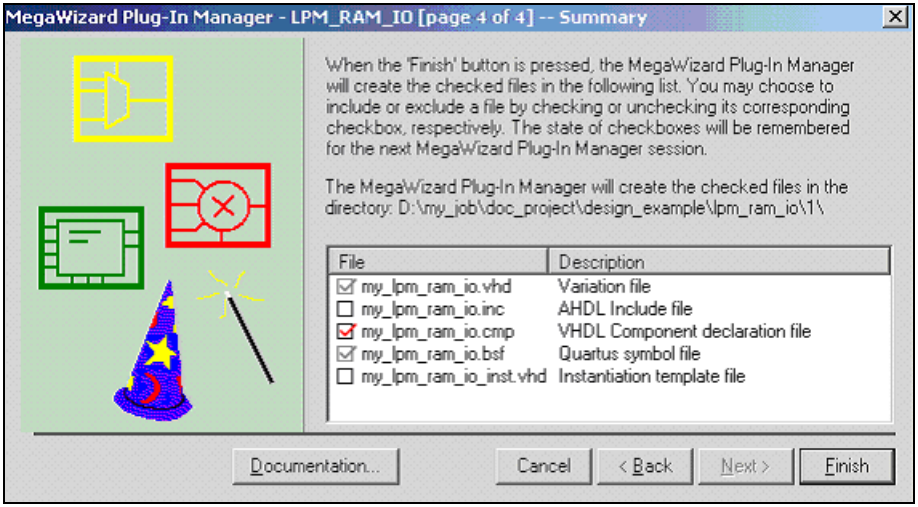
Figure 2–41. MegaWizard Plug-In Manager - LPM_RAM_IO [page 3 of 4]



20. On page 4, turn on the options corresponding to a **VHDL** design file(**.vhd**), Quartus symbol file, and VHDL component file, so they can be created for the project (Figure 2–42).

21. Click Next.

Figure 2–42. MegaWizard Plug-In Manager LPM_RAM_IO [page 4 of 4] Summary



22. Click **Finish**. The `lpm_ram_io` megafunction is built.

23. Move the mouse to place the RAM symbol in between the input and output ports of the **top.bdf** file. Click the left mouse button to place the RAM symbol. You have now completed the design file.

24. Select **Save** (**File** menu) to save the design.

Implement the lpm_ram_io Single-port Memory

Next, assign the EP2A15B724C7 device to the project and compile the project.

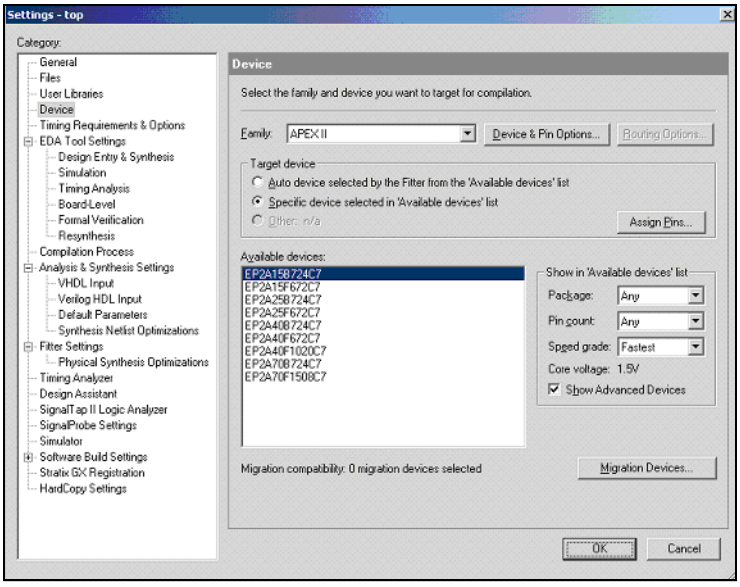
1. Select **Settings** (**Assignments** menu) to open the **Settings** dialog box.

2. Click the **Devices** category. Ensure that APEX II is selected in the Family field (Figure 2–43).

3. Select **EP2A15B724C7** under the '**Available devices**' list in the **Target device** section.

4. Click **OK**.

Figure 2–43. Settings - top

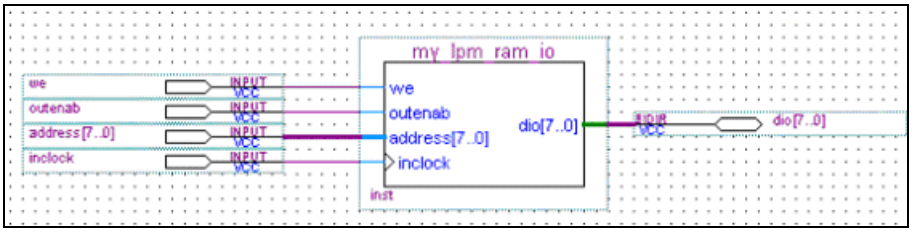


5. Select **Start Compilation** (**Processing** menu) to compile the design.

6. Click **OK** when the **Full compilation was successful** message box appears.

7. In the **Compilation Report** window, you can expand the Fitter section and click on the Floorplan View option to view how the module is implemented in the APEX II device.

Figure 2–44. Top-level Design for Synchronous Write/Read Cycle

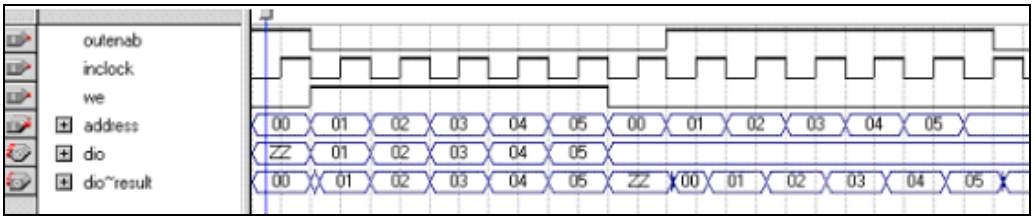


Functional Results - Simulate the lpm_ram_io Single-port Memory

Finally, simulate the design to verify the results. Setup the Quartus II Simulator by performing the following steps:

1. From the Processing menu, select **Generate Functional Simulation Netlist**.
2. Click **OK**.
3. Select **Wizards > Simulator Setting Wizard** (**Assignments** menu).
4. In the Category section, highlight the **Simulator** category.
5. Select **Functional** under the **Simulation mode**.
6. Select **Start Simulation** (**Processing** menu) to start the simulation.
7. Click **OK** when the **Simulation was successful** message box appears.
8. In the **Simulation Report** window, look at the simulation output waveform and verify the results. Figure 2–45 shows the expected simulation results.

Figure 2–45. Functional Wave Form for Synchronous Write/Read Cycle



When the single-port memory is configured in other modes, the steps to run the project are similar. For the additional modes, just the top-level schematic and functional simulation waveforms are illustrated below.

Figure 2–46. Top-level Design for Asynchronous Write/Read Cycle

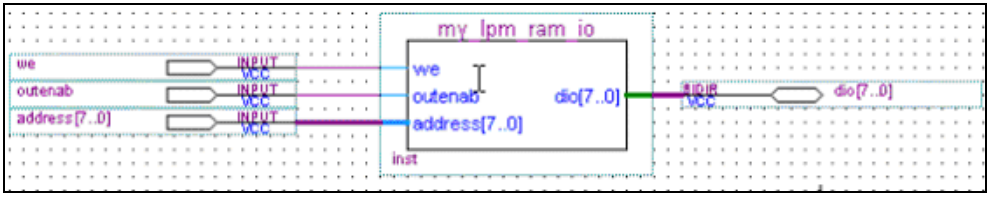
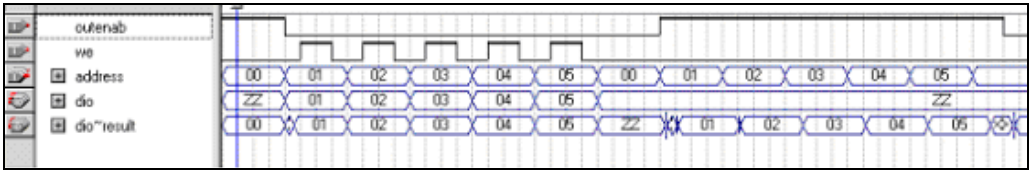


Figure 2–47. Functional Wave Form for Asynchronous Write/Read Cycle



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.

Chapter 3. Specifications

Ports & Parameters for RAM Megafunctions

The Quartus® II software provides three megafunctions that support single-port and dual-port RAM functionality: `lpm_ram_dp`, `lpm_ram_dq`, and `lpm_ram_io`. This chapter describes the ports and parameters for the RAM megafunctions.

The parameter details are relevant only 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.

Ports and Parameters for the lpm_ram_dp Megafunction

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

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

Name	Required	Description	Comment
`wren`	Yes	Write enable input.	
`Data[]`	Yes	Data input to memory.	Input port `LPM_WIDTH` wide.
`Wraddress[]`	Yes	Write address to memory.	Input port `LPM_WIDTHAD` wide.
`wrclock`	No	Rising edge triggered clock for write operation.	In use, the `wrclock` port acts as the clock for the write operation and functions as the clock signal to any registers present on the `wraddress`, `wren`, and `data[]` ports. For a single-clock synchronous design, you can tie `rdclock` and `wrclock` together. In ACEX 1K, APEX 20KC, APEX 20KE, APEX II, Excalibur, FLEX 10KE, and Mercury devices, the write into the internal memory array occurs during the next low period of the clock.

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

Name	Required	Description	Comment
wrclken	No	Clock enable for wrclock.	This port is used by all registers clocked by wrclock. For single-clock synchronous designs, you can tie rdclock and wrclock together.
rden	No	Read enable input. Disables reading when low (0). The default is 1.	In ACEX 1K, APEX 20KC, APEX 20KE, APEX II, Excalibur, FLEX 6000, FLEX 10KE, and Mercury devices, the rden port controls a latch that remembers the value last read while the rden port was high. In APEX 20K devices, the rden port becomes a power down signal. When using simulation models for other EDA simulators, the rden port always behaves as if it is targeting an ACEX 1K, APEX 20KC, APEX 20KE, APEX II, Excalibur, FLEX 6000, FLEX 10KE, or Mercury device.
Rdaddress[]	Yes	Read address input to the memory.	Input port LPM_WIDTHAD wide.
rdclock	No	Rising edge triggered clock for read operation.	This port is used for registered read ports, such as q, rdaddress, and rden. In use, the rdclock port acts as the clock for read operation and functions as the clock signal to any registers present on the rdaddress, rden, and q[] ports.
rdclken	No	Clock enable for rdclock.	This port is used by all registers clocked by rdclock. For single-clock synchronous designs, you can tie rdclock and wrclock together.

Table 3–2. lpm_ram_dp Megafunction Output Ports

Name	Required	Description	Comment
Q[]	Yes	Data output from the memory.	Output port LPM_WIDTH wide.

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

Name	Type	Required	Comment
LPM_WIDTH	Integer	Yes	Specifies the width of the data[] and q[] ports.
LPM_WIDTHAD	Integer	Yes	Width of the rdaddress[] and wraddress[] ports.
LPM_NUMWORDS	Integer	No	Number of words stored in memory. This value must be within the range 2 ^ WIDTHAD-1 < NUMWORDS <= 2 ^ WIDTHAD. If omitted, the default is 2 ^ WIDTHAD.
LPM_INDATA	String	No	Determines the clock used by the data[] port. Values are "UNREGISTERED" and "REGISTERED." The default is "REGISTERED."
LPM_OUTDATA	String	No	Determines the clock used by the q[] port. Values are "UNREGISTERED" and "REGISTERED." The default is "REGISTERED."
LPM_RDADDRESS_CONTROL	String	No	Determines the clock used by the rdaddress and rden ports. Values are "UNREGISTERED" and "REGISTERED." The default is "REGISTERED."

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

Name	Type	Required	Comment
LPM_FILE	String	No	Name of the Memory Initialization File (**.mif**) or Hexadecimal (Intel-Format) File (**.hex**) containing RAM initialization data ("<file name>"), or "UNUSED." The default is "UNUSED." If omitted, the contents default to all zeros. The wren port must be registered to support memory initialization.
USE_EAB	String	No	Altera-specific parameter. Values are "ON", "OFF", and "UNUSED." Setting the USE_EAB parameter to OFF, prevents the Quartus II software from using Embedded System Blocks (ESBs) to implement the logic in ESBs. To implement the logic in APEX 20K, APEX II, Excalibur, and Mercury devices or EABs in ACEX 1K and FLEX 10KE devices, it can only use flipflops or latches. (The "ON" setting is not useful in memory functions: the Quartus II software automatically implements memory functions in ESBs or EABs by default.) This parameter is not available for simulation with other EDA simulators and for FLEX 6000, MAX 3000, and MAX 7000 devices. If you want to use this parameter when you instantiate the function in a Block Design File (**.bdf**), you must specify it by entering the parameter name and value manually with the Parameters tab (Symbol Properties Command) or the Parameters tab (Block Properties Command). You can also use this parameter name in a Text Design File (**.tdf**) or a Verilog Design File (**.v**). You must use the LPM_HINT parameter to specify the USE_EAB parameter in VHDL Design Files.

Ports and Parameters for the lpm_ram_dq Megafunction

Table 3–4 shows the input ports, Table 3–5 shows the output ports, and Table 3–6 shows the lpm_ram_dq megafunction parameters.

Table 3–4. lpm_ram_dq Megafunction Input Ports

Name	Required	Description	Comment
data[]	Yes	Data input to memory.	Input port LPM_WIDTH wide.
address[]	Yes	Address input to the memory.	Input port LPM_WIDTHAD wide.
we	No	Write enable input. Enables write operations to the memory when high.	Required if inclock is not present. If only the we port is used, the data on the address[] port should not change while we is high. If the data on the address[] port changes while the we port is high, all memory locations that are addressed are overwritten with data[].
inclock	No	Synchronizes memory loading.	If the inclock port is used, the we port acts as an enable for write operations synchronized to the rising edge of the inclock signal. If the inclock port is not used, the we port acts as an enable for asynchronous write operations. In addition, if the inclock port is not used, the LPM_INDATA and LPM_ADDRESS_CONTROL parameters must be set to "UNREGISTERED."
outclock	No	Synchronizes q outputs from memory.	The addressed memory content-to-q[] response is synchronous when the outclock port is connected and asynchronous when it is not connected. In addition, if the outclock port is not used, the LPM_OUTDATA parameter must be set to "UNREGISTERED."

Table 3–5. lpm_ram_dq Megafunction Output Ports

Name	Required	Description	Comment
q[]	Yes	Data output from the memory.	Output port LPM_WIDTH wide.

Table 3–6. lpm_ram_dq Megafunction Parameters (Part 1 of 2)

Name	Type	Required	Comment
LPM_WIDTH	Integer	Yes	Specifies the width of the data[] and q[] ports.
LPM_WIDTHAD	Integer	Yes	Width of the address[] ports. LPM_WIDTHAD should be (but is not required to be) equal to log2 (LPM_NUMWORDS). If LPM_WIDTHAD is too small, some memory locations will not be addressable. If it is too large, the addresses that are too high will return undefined (X) logic levels.
LPM_NUMWORDS	Integer	No	Number of words stored in memory. If omitted, the default is 2 ^ LPM_WIDTHAD. In general, this value should be (but is not required to be) 2 ^ LPM_WIDTHAD-1 < LPM_NUMWORDS <= 2 ^ LPM_WIDTHAD. If omitted, the default is 2 ^ LPM_WIDTHAD.
LPM_FILE	String	No	Name of the Memory Initialization File (**.mif**) or Hexadecimal (Intel-Format) Output File (**.hexout**) containing ROM initialization data ("<file name>"), or "UNUSED." The default is "UNUSED." If omitted, the contents default to all zeros. The we port must be registered to support memory initialization.
LPM_INDATA	String	No	Values are "REGISTERED", "UNREGISTERED" and "UNUSED." Controls whether the data port is registered. If omitted, the default is "REGISTERED." If the inclock port is not used, the LPM_INDATA and LPM_ADDRESS_CONTROL parameters must be set to "UNREGISTERED."
LPM_ADDRESS_CONTROL	String	No	Specifies whether the address and we ports are registered. Values are "REGISTERED", "UNREGISTERED", and "UNUSED." If omitted, the default is "REGISTERED." If LPM_ADDRESS_CONTROL is "UNREGISTERED", the we port is level-sensitive, so that when the we port is high, the address[] port must be stable to prevent other memory locations from being overwritten. If the inclock port is not used, the LPM_INDATA and LPM_ADDRESS_CONTROL parameters must be set to "UNREGISTERED."

Table 3–6. lpm_ram_dq Megafunction Parameters (Part 2 of 2)

Name	Type	Required	Comment
LPM_OUTDATA	String	No	Specifies whether the q and internal eq ports are registered. Values are "REGISTERED", "UNREGISTERED" and "UNUSED." If omitted, the default is "REGISTERED." If the inclock port is not used, the LPM_OUTDATA parameter must be set to "UNREGISTERED."
LPM_HINT	String	No	Lets you 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.
USE_EAB	String	No	Altera-specific parameter. Values are "ON", "OFF", and "UNUSED". Setting the USE_EAB parameter to OFF, prevents the Quartus II software from using Embedded System Blocks (ESBs) to implement the logic in APEX 20K, APEX II, Excalibur, and Mercury devices, or EABs in ACEX 1K and FLEX 10K devices; it can only use flipflops or latches. (The "ON" setting is not useful in memory functions: the Quartus II software automatically implements memory functions in ESBs or EABs by default.) This parameter is not available for simulation with other EDA simulators and for FLEX 6000, MAX 3000, and MAX 7000 devices. If you want to use this parameter when you instantiate the function in a Block Design File (**.bdf**), you must specify it by entering the parameter name and value manually with the Parameters tab (Symbol Properties Command) or the Parameters tab (Block Properties Command). You can also use this parameter name in a Text Design File (**.tdf**) or a Verilog Design File (**.v**). You must use the LPM_HINT parameter to specify the USE_EAB parameter in VHDL Design Files.

Ports and Parameters for the lpm_ram_io Megafunction

Table 3–7 shows the input ports, Table 3–8 shows the output ports, and Table 3–9 shows the lpm_ram_dp megafunction parameters.

Table 3–7. lpm_ram_io Megafunction Input Ports

Name	Required	Description	Comment
address[]	Yes	Address input to the memory.	Input port LPM_WIDTHAD wide. If memenab is used, it must be inactive when address[] is changing.
we	Yes	Write enable input. Enables write operations to the memory when high.	If no clock ports are used, the data on the address[] port must not change when we is high (1). This port is required if clock is not present.
inclock	No	Synchronizes memory loading.	If the inclock port is used, the we port acts as an enable for write operations synchronized to the rising edge of the inclock signal. If the inclock port is not used, the we port acts as an enable for asynchronous write operations.
outclock	No	Synchronizes dio[] from memory.	The addressed memory content-to-q[] response is synchronous when the outclock port is connected and asynchronous when it is not connected.
memenab	No	Memory output tri-state enable.	Either memenab or outenab must be connected. If memenab is present, it must be inactive when address[] is changing. This port is available for backward compatibility only. Altera recommends that you not use this port.
outenab	No	Output Enable input. High (1): dio[] from Memory [address] Low (0): Memory [address] from dio[].	Either memenab or outenab must be present.

Table 3–8. lpm_ram_io Megafunction Output Ports

Name	Required	Description	Comment
dio[]	Yes	Bidirectional data port for the memory.	Bidirectional port LPM_WIDTH wide.

Table 3–9. lpm_ram_io Megafunction Parameters (Part 1 of 2)

Name	Type	Required	Comment
LPM_WIDTH	Integer	Yes	Specifies the width of the dio[], internal data, and q[] ports.
LPM_WIDTHAD	Integer	Yes	Width of the address[] ports. LPM_WIDTHAD should be (but is not required to be) equal to log2 (LPM_NUMWORDS). If LPM_WIDTHAD is too small, some memory locations will not be addressable. If it is too large, the addresses that are too high will return undefined (X) logic levels.
LPM_NUMWORDS	Integer	No	Number of words stored in memory. In general, this value should be (but is not required to be) 2 ^ LPM_WIDTHAD-1 < LPM_NUMWORDS <= 2 ^ LPM_WIDTHAD. If omitted, the default is 2 ^ LPM_WIDTHAD.
LPM_FILE	String	No	Name of the Memory Initialization File (**.mif**) or Hexadecimal (Intel-Format) Output File (**.hexout**) containing ROM initialization data ("<file name>"), or "UNUSED." The default is "UNUSED." If omitted, the contents default to all zeros. The we port must be registered to support memory initialization.
LPM_INDATA	String	No	Specifies whether the data port is registered. Values are "REGISTERED", "UNREGISTERED" and "UNUSED." If omitted, the default is "REGISTERED."
LPM_ADDRESS_CONTROL	String	No	Specifies whether the address, memenab, and we ports are registered. Values are "REGISTERED", "UNREGISTERED", and "UNUSED." If omitted, the default is "REGISTERED."
LPM_OUTDATA	String	No	Specifies whether the dio[] port is registered. Values are "REGISTERED", "UNREGISTERED" and "UNUSED." If omitted, the default is "REGISTERED."
LPM_HINT	String	No	Lets you specify Altera-specific parameters in VHDL Design Files (**.vhd**). The default is "UNUSED."

Table 3–9. lpm_ram_io Megafunction Parameters (Part 2 of 2)

Name	Type	Required	Comment
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.
USE_EAB	String	No	Altera-specific parameter. Values are "ON", "OFF", and "UNUSED." Setting the USE_EAB parameter to OFF, prevents the Quartus II software from using Embedded System Blocks (ESBs) to implement the logic in APEX 20K, APEX II, Excalibur, and Mercury devices, or EABs in ACEX 1K and FLEX 10K devices; it can only use flipflops or latches. (The "ON" setting is not useful in memory functions: the Quartus II software automatically implements memory functions in ESBs or EABs by default.) This parameter is not available for simulation with other EDA simulators and for FLEX 6000, MAX 3000, and MAX 7000 devices. If you want to use this parameter when you instantiate the function in a Block Design File (**.bdf**), you must specify it by entering the parameter name and value manually with the Parameters tab (Symbol Properties Command) or the Parameters tab (Block Properties Command). You can also use this parameter name in a Text Design File (**.tdf**) or a Verilog Design File (**.v**). You must use the LPM_HINT parameter to specify the USE_EAB parameter in VHDL Design Files.