



# DDR and DDR2 SDRAM

---

## High-Performance Controller User Guide



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
[www.altera.com](http://www.altera.com)

MegaCore Version:	6.1
Document Version:	6.1
Document Date:	December 2006

Copyright © 2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



UG-01010-1.0



## About This User Guide

Revision History .....	v
How to Contact Altera .....	vi
Typographic Conventions .....	vii

## Chapter 1. About These MegaCore Functions

Release Information .....	1-1
Device Family Support .....	1-1
Features .....	1-2
General Description .....	1-2
OpenCore Plus Evaluation .....	1-3
Performance .....	1-4

## Chapter 2. Getting Started

Design Flow .....	2-1
DDR & DDR2 SDRAM High-Performance Controller Walkthrough .....	2-2
Create a New Quartus II Project .....	2-3
Launch the MegaWizard Plug-In Manager .....	2-4
Set Up Simulation .....	2-10
Simulate the Example Design .....	2-14
Simulating With the ModelSim Simulator .....	2-15
Simulating With Other Simulators .....	2-16
Compile the Example Design .....	2-20
Program a Device .....	2-22
Implement Your Design .....	2-23
Set Up Licensing .....	2-23

## Chapter 3. Specifications

Functional Description .....	3-1
Control Logic .....	3-1
OpenCore Plus Time-Out Behavior .....	3-3
Example Design .....	3-3
Interfaces & Signals .....	3-5
Interface Description .....	3-5
Signals .....	3-16
Parameters .....	3-18
MegaCore Verification .....	3-19
Simulation Testing .....	3-19
Hardware Testing .....	3-19





# About This User Guide

## Revision History

The table below displays the revision history for the chapters in this user guide.

Date	Version	Changes Made
December 2006	6.1	First 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](http://www.altera.com). For technical support on this product, go to [www.altera.com/mysupport](http://www.altera.com/mysupport). For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	<a href="http://www.altera.com/mysupport/">www.altera.com/mysupport/</a>	<a href="http://www.altera.com/mysupport/">www.altera.com/mysupport/</a>
	800-800-EPLD (3753) 7:00 a.m. to 5:00 p.m. Pacific Time	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	<a href="http://www.altera.com">www.altera.com</a>	<a href="http://www.altera.com">www.altera.com</a>
Altera literature services	<a href="mailto:literature@altera.com">literature@altera.com</a>	<a href="mailto:literature@altera.com">literature@altera.com</a>
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	<a href="ftp://ftp.altera.com">ftp.altera.com</a>	<a href="ftp://ftp.altera.com">ftp.altera.com</a>

## Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
<b>Bold Type with Initial Capital Letters</b>	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: <b>Save As</b> dialog box.
<b>bold type</b>	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: <b>f<sub>MAX</sub></b> , <b>lqdesigns</b> directory, <b>d:</b> drive, <b>chiptrip.gdf</b> file.

Visual Cue	Meaning
<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: $t_{PIA}$ , $n + 1$ .  Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <file name>, <project name>.pdf 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.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.



# 1. About These MegaCore Functions

## Release Information

Table 1–1 provides information about this release of the DDR and DDR2 SDRAM High-Performance Controller MegaCore® functions.

<b>Table 1–1. DDR &amp; DDR2 SDRAM High-Performance Controller Release Information</b>	
<b>Item</b>	<b>Description</b>
Version	6.1
Release Date	December 2006
Ordering Codes	IP-SDRAM/HPDDR (DDR SDRAM) IP-SDRAM/HPDDR2 (DDR2 SDRAM)
Product IDs	00BE (DDR SDRAM) 00BF (DDR2 SDRAM) 00CO (ALTMEMPHY Megafunction)
Vendor ID	6AF7

## Device Family Support

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

- *Full support* means the MegaCore function meets all functional and timing requirements for the device family and may be used in production designs
- *Preliminary support* means the MegaCore function 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 1–2 shows the level of support offered by the DDR and DDR2 SDRAM High-Performance Controller to each of the Altera device families.

<b>Table 1–2. Device Family Support</b>		
<b>Device Family</b>	<b>Support</b>	
	<b>DDR SDRAM</b>	<b>DDR2 SDRAM</b>
Stratix II	Full	Full
Stratix II GX	Full	Full
Stratix III	Preliminary	Preliminary
Other device families	No support	No support

## Features

- Support for ALTMEMPHY megafunction
- Support for industry-standard DDR and DDR2 SDRAM devices and modules
- Optional user-controller refresh
- Optional Avalon® Memory-Mapped local interface
- Support for OpenCore Plus evaluation
- Easy-to-use MegaWizard® interface
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators

## General Description

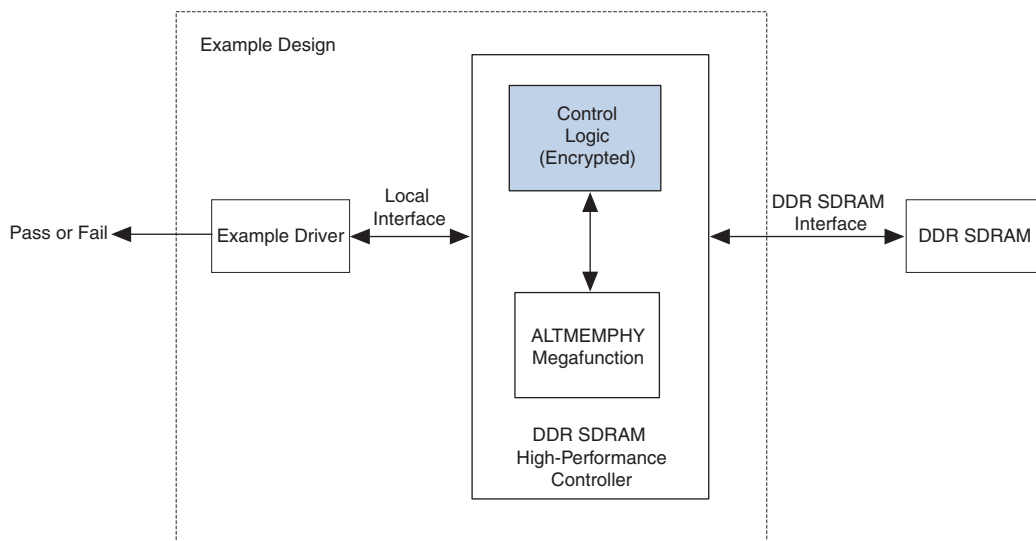
The Altera DDR and DDR2 SDRAM High-Performance Controller MegaCore functions provide simplified interfaces to industry-standard DDR SDRAM and DDR2 SDRAM. The MegaCore functions work in conjunction with the Altera ALTMEMPHY megafunction.



For more information on the ALTMEMPHY megafunction, refer to Quartus II Help.

Figure 1–1 shows a system-level diagram including the example design that the DDR or DDR2 SDRAM Controller MegaCore functions create for you.



**Figure 1–1. DDR & DDR2 SDRAM Controller System-Level Diagram**

**Note to Figure 1–1:**

(1) Optional, for Stratix series and HardCopy II devices only.

The MegaWizard Plug-In Manager generates an example design, instantiates a phase-locked loop (PLL), an example driver, your DDR or DDR2 SDRAM controller custom variation, and an optional DLL (for Stratix series only). The example design is a fully-functional design that can be simulated, synthesized, and used in hardware. The example driver is a self-test module that issues read and write commands to the controller and checks the read data to produce the pass/fail and test complete signals.

## OpenCore Plus Evaluation

With Altera's free OpenCore Plus evaluation feature, you can perform the following actions:

- Simulate the behavior of a megafunction (Altera MegaCore function or AMPP<sup>SM</sup> megafunction) within your system
- Verify the functionality of your design, as well as evaluate its size and speed quickly and easily
- Generate time-limited device programming files for designs that include MegaCore functions
- Program a device and verify your design in hardware

You only need to purchase a license for the megafunction when you are completely satisfied with its functionality and performance, and want to take your design to production.



For more information on OpenCore Plus hardware evaluation using the DDR and DDR2 SDRAM Controller, see “[OpenCore Plus Time-Out Behavior](#)” on [page 3–3](#) and *Application Note 320: OpenCore Plus Evaluation of Megafunctions*.

## Performance

[Table 1–3](#) shows typical performance results for the DDR SDRAM controller using the Quartus® II software version 6.1.

<b>Table 1–3. Typical Performance</b>		
Device	System $f_{\text{MAX}}$ (MHz)	
	DDR SDRAM	DDR2 SDRAM
Stratix II	200	333
Stratix III	200 (1)	400 (1)

**Note to [Table 1–3](#):**

(1) Pending device characterization.



For more information on device performance, see the relevant device handbook.

[Table 1–4](#) shows typical sizes for the DDR SDRAM controller on Stratix II devices.

<b>Table 1–4. Typical Size—Stratix II Devices</b>				
Local Data Width (Bits)	Memory Width (Bits)	Combinational ALUTs	Logic Registers	Memory Blocks (M4K)
32	8	1,420	1,480	3
64	16	1,530	1693	6
256	64	2,136	2,914	20
288	72	2,291	3,119	22

Table 1–5 shows typical sizes for the DDR SDRAM controller on Stratix III devices (with no calibration logic).

<b>Table 1–5. Typical Size—Stratix III Devices</b>				
<b>Local Data Width (Bits)</b>	<b>Memory Width (Bits)</b>	<b>Combinational ALUTs</b>	<b>Logic Registers</b>	<b>Memory Blocks (MLAB)</b>
32	8	796	1,163	5
64	16	937	1,440	10
256	64	1,773	3,110	39
288	72	1,907	3,383	44



### Design Flow

To evaluate the DDR and DDR2 SDRAM High-Performance Controller MegaCore® functions using the OpenCore Plus feature, include these steps in your design flow:

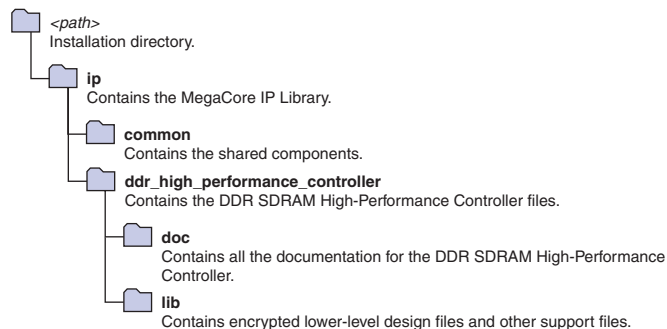
1. Obtain and install the DDR and DDR2 SDRAM High-Performance Controller MegaCore.

The DDR and DDR2 SDRAM High-Performance Controller MegaCore functions are part of the MegaCore IP Library, which is distributed with the Quartus® II software and downloadable from the Altera® website, [www.altera.com](http://www.altera.com).

For system requirements and installation instructions, refer to *Quartus II Installation & Licensing for Windows* or *Quartus II Installation & Licensing for UNIX & Linux* on the Altera website at [www.altera.com/literature/lit-qts.jsp](http://www.altera.com/literature/lit-qts.jsp).

Figure 2–1 shows the directory structure after you install the DDR and DDR2 SDRAM High-Performance Controller MegaCore functions, where *<path>* is the installation directory. The default installation directory on Windows is `c:\altera\61`; on UNIX and Solaris it is `/opt/altera/61`.

**Figure 2–1. Directory Structure**



2. Create a custom variation of the DDR or DDR2 SDRAM High-Performance Controller MegaCore function.

3. Use the IP functional simulation model to verify the operation of the example design and the example driver.



For more information on IP functional simulation models, see the *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

4. Use the Quartus II software to add constraints to the example design, compile the example design, and perform post-compilation timing analysis.
5. Perform gate-level timing simulation, or if you have a suitable development board, you can generate an OpenCore Plus time-limited programming file, which you can use to verify the operation of the example design in hardware.
6. Purchase a license for the DDR or DDR2 SDRAM High-Performance Controller MegaCore function.

After you have purchased a license for the DDR and DDR2 SDRAM High-Performance Controller, follow these additional steps:

1. Set up licensing.
2. Generate a programming file for the Altera device(s) on your board.
3. Program the Altera device(s) with the completed design.

## DDR & DDR2 SDRAM High- Performance Controller Walkthrough

This walkthrough explains how to create a custom variation of the DDR or DDR2 SDRAM Controller MegaCore function using the MegaWizard® Plug-In Manager and the Quartus II software. As you go through the wizard, each step is described in detail.

This walkthrough requires the following steps:

- “Create a New Quartus II Project” on page 2–3
- “Launch the MegaWizard Plug-In Manager” on page 2–4
- “Parameterize” on page 2–7
- “Set Up Simulation” on page 2–10
- “Generate Files” on page 2–11

## Create a New Quartus II Project

You need to create a new Quartus II project with the **New Project Wizard**, which specifies the working directory for the project, assigns the project name, and designates the name of the top-level design entity. To create a new project follow these steps:

1. Choose **Programs > Altera > Quartus II <version>** (Windows Start menu) to run the Quartus II software. Alternatively, you can use the Quartus II Web Edition software.
2. Choose **New Project Wizard** (File menu).
3. Click **Next** in the **New Project Wizard Introduction** page (the introduction page does not display if you turned it off previously).
4. In the **New Project Wizard: Directory, Name, Top-Level Entity** page, enter the following information:
  - a. Specify the working directory for your project. For example, this walkthrough uses the **c:\altera\projects\ddr\_project** directory.
  - b. Specify the name of the project. This walkthrough uses **project** for the project name.




The Quartus II software automatically specifies a top-level design entity that has the same name as the project. Do not change it.

5. Click **Next** to close this page and display the **New Project Wizard: Add Files** page.




When you specify a directory that does not already exist, a message asks if the specified directory should be created. Click **Yes** to create the directory.

6. If you installed the MegaCore IP Library in a different directory from where you installed the Quartus II software, you must add the user libraries:
    - a. Click **User Libraries**.
    - b. Type `<path>\ip` into the **Library name** box, where `<path>` is the directory in which you installed the DDR and DDR2 SDRAM High-Performance Controller.
    - c. Click **Add to** add the path to the Quartus II project.
    - d. Click **OK** to save the library path in the project.
  7. Click **Next** to close this page and display the **New Project Wizard: Family & Device Settings** page.
  8. On the **New Project Wizard: Family & Device Settings** page, choose the target device family in the **Family** list.
-  The DDR and DDR SDRAM controllers support Stratix II, Stratix II GX, and Stratix III devices.
9. The remaining pages in the **New Project Wizard** are optional. Click **Finish** to complete the Quartus II project.

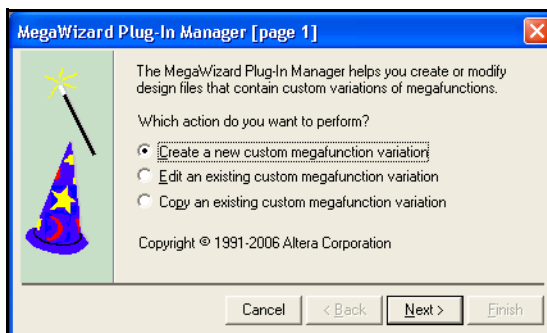
You have finished creating your new Quartus II project.

### Launch the MegaWizard Plug-In Manager

To launch the MegaWizard Plug-In Manager in the Quartus II software, follow these steps:

1. Start the MegaWizard Plug-In Manager by choosing the **MegaWizard Plug-In Manager** command (Tools menu). The **MegaWizard Plug-In Manager** dialog box displays (see [Figure 2–2](#)).
-  Refer to Quartus II Help for more information on how to use the MegaWizard Plug-In Manager.

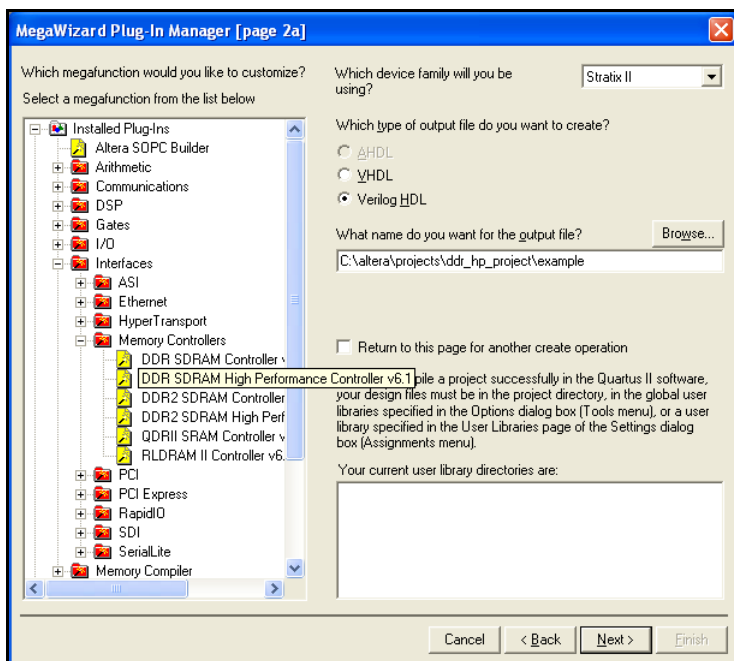


**Figure 2–2. MegaWizard Plug-In Manager**

2. Specify that you want to create a new custom megafunction variation and click **Next**.
3. Expand the **Interfaces > Memory Controllers** directory then click either **DDR SDRAM High-Performance Controller v6.1** or **DDR2 SDRAM High-Performance Controller v6.1**.
4. Select the output file type for your design; the wizard supports VHDL and Verilog HDL.
5. The MegaWizard Plug-In Manager shows the project path that you specified in the **New Project Wizard**. Append a variation name for the MegaCore function output files *<project path>\<variation name>*. [Figure 2–3 on page 2–6](#) shows the wizard after you have made these settings.



The *<variation name>* must be a different name from the project name and the top-level design entity name.

**Figure 2–3. Select the MegaCore Function**

6. Click **Next** to display the **Parameter Settings** page for the DDR and DDR2 SDRAM High-Performance Controller MegaCore functions (see [Figure 2–4](#)).



You can change the page that the MegaWizard Plug-In Manager displays by clicking **Next** or **Back** at the bottom of the dialog box. You can move directly to a named page by clicking the **Parameter Settings**, **Simulation Model**, or **Summary** tab.

Also, you can directly display individual parameter settings by clicking on the **Memory Settings**, **PHY Settings**, or **Controller Settings** tabs.

Figure 2–4. Parameters

**example Settings**

**MegaCore®** **DDR SDRAM High Performance Controller** **Version 6.1** [About] [Documentation]

1 Parameter Settings 2 Simulation Model 3 Summary

Memory Settings > PHY Settings > Controller Settings >

**General Settings**

Device family:

Speed grade: 4

PLL reference clock frequency: 250 MHz (4000 ps)

Memory clock frequency: 250 MHz (4000 ps)

Local interface clock frequency: 125 MHz (8000 ps)

**Show in 'Memory Presets' List**

Parameter	Value
Memory vendor	(All)
Memory format	(All)
Maximum frequency supported ...	(All)

Show All

**Memory Presets**

Presets

- JEDEC DDR-400 128Mb x8
- JEDEC DDR-400 256Mb x8
- JEDEC DDR-400 512Mb x8
- Micron MT9VDDT3274AG-40B
- Infineon HYB25D25616OBT-5
- Micron MT46V16M16TG-5B

Selected preset: JEDEC DDR-400 128Mb x8 [Modify parameters...]

Description: DDR SDRAM 16 MB 8 bits wide Discrete Device CAS 3.0 x 1

[Cancel] [Back] [Next] [Finish]

### Parameterize

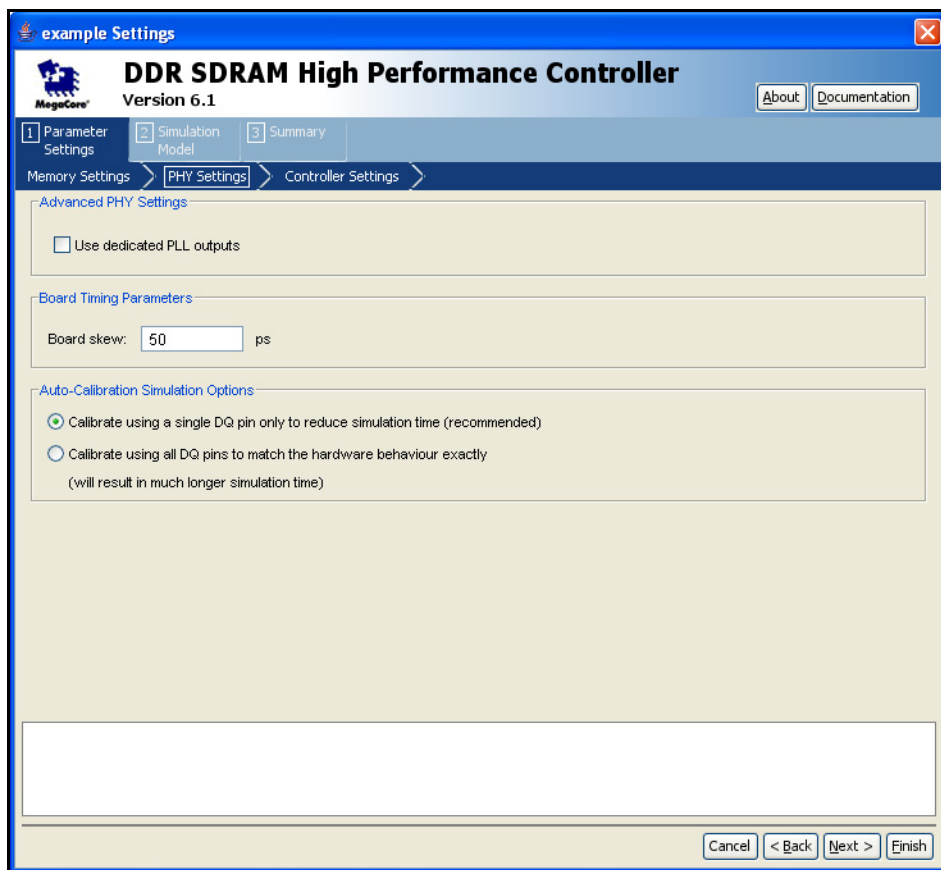
To parameterize your MegaCore function, follow these steps:



For more information on the memory and PHY settings, refer to Quartus II Help.

1. Select your memory settings.
2. Click the **PHY Settings** tab (see [Figure 2–5 on page 2–8](#)).

**Figure 2–5. PHY Settings**

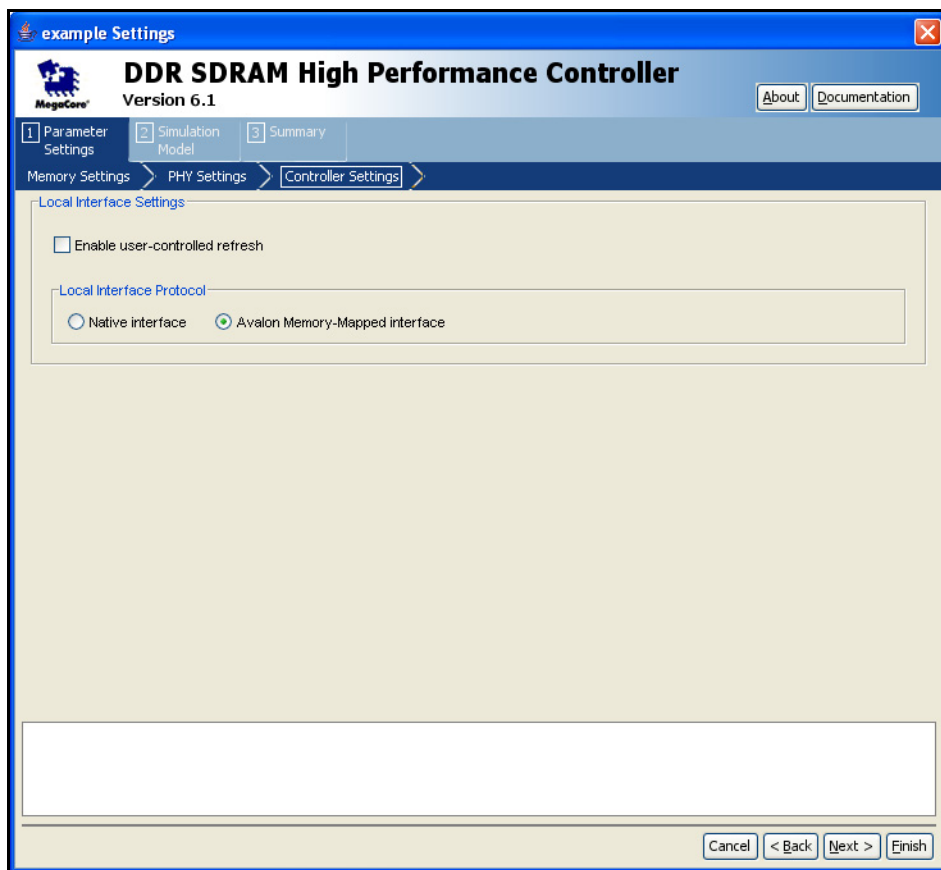


3. Select your PHY settings.
4. Click the **Controller Settings** tab (see [Figure 2–6](#)).



For more information on controller settings, see [“Controller Settings” on page 3–19](#).

Figure 2–6. Controller Parameters

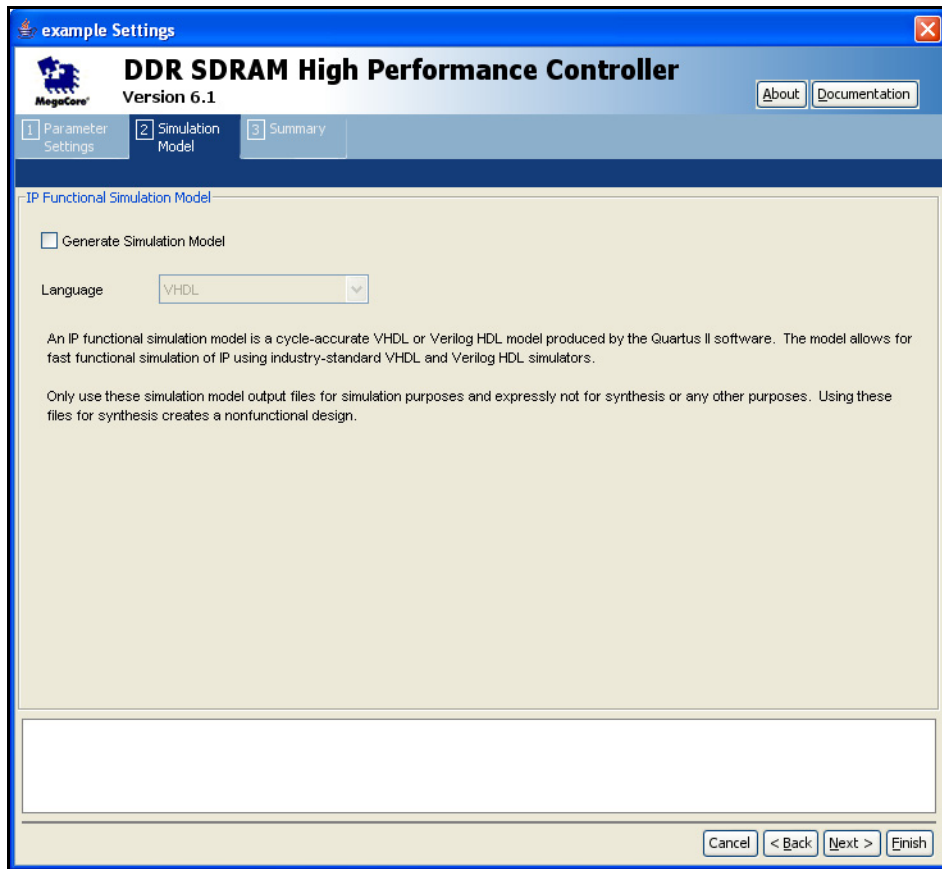


5. Choose your local interface settings.
6. Select **Native** or **Avalon Memory-Mapped** interface. The Avalon® Memory-Mapped (MM) interface allows you to easily connect to other Avalon-MM peripherals.



For more information on the Avalon-MM interface, see the *Avalon Memory-Mapped Interface Specification*.

7. Click **Next** (or the **Simulation Model** tab) to display the simulation setup page (see Figure 2–7).

**Figure 2–7. Simulation Model**

## Set Up Simulation

An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software. The model allows for fast functional simulation of IP using industry-standard VHDL and Verilog HDL simulators.

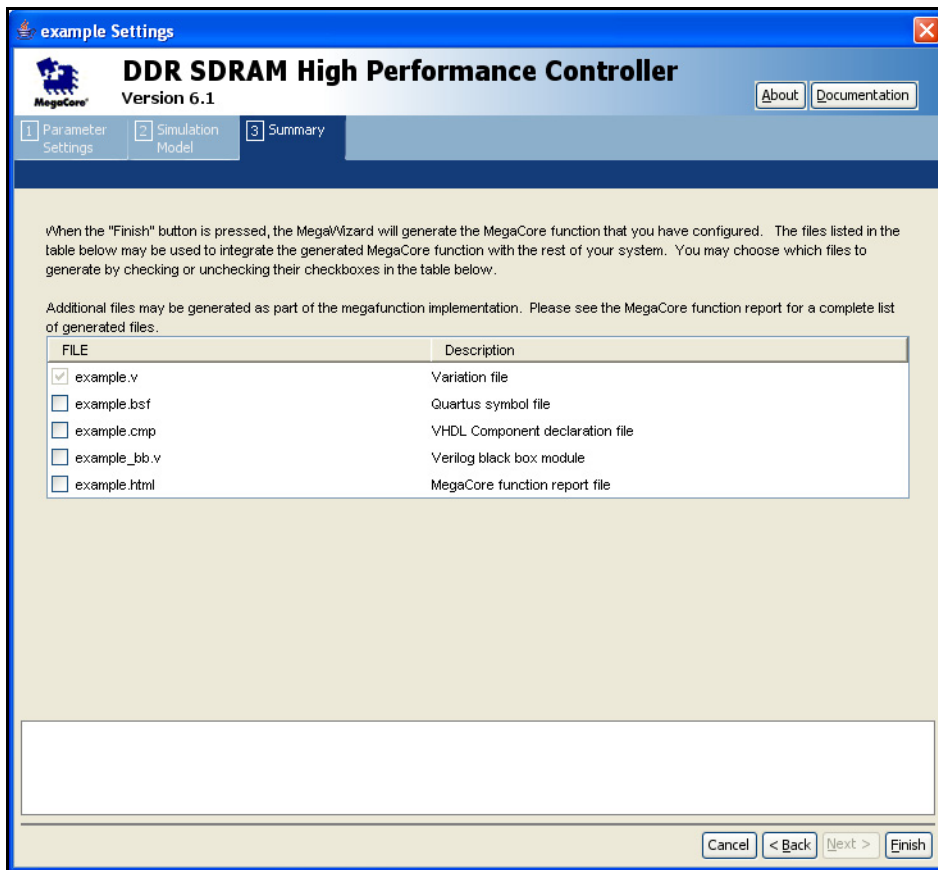


You may only use these models for simulation and expressly not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

To generate an IP functional simulation model for your MegaCore function, follow these steps:

1. Turn on **Generate Simulation Model**.
2. Choose the language from the **Language** list.
3. Click **Next** (or the **Summary** tab) to display the summary page (see [Figure 2–8](#)).

**Figure 2–8. Summary**



### *Generate Files*

You can use the check boxes on the **Summary** page to enable or disable the generation of specified files. A gray checkmark indicates a file that is automatically generated; a red checkmark indicates an optional file.

You can click **Back** to display the previous page or click **Parameters Setting**, **Simulation Library** or **Summary**, if you want to change any of the MegaWizard options.

To generate the files, follow these steps:

1. Turn on the files you wish to generate.



At this stage you can still click **Back** or the pages to display any of the other pages in the MegaWizard Plug-In Manager, if you want to change any of the parameters.

2. To generate the specified files and close the MegaWizard Plug-In Manager, click **Finish**.



The generation phase may take several minutes to complete.

Figure 2–9 shows the generation report.

**Figure 2–9. Generation Report**

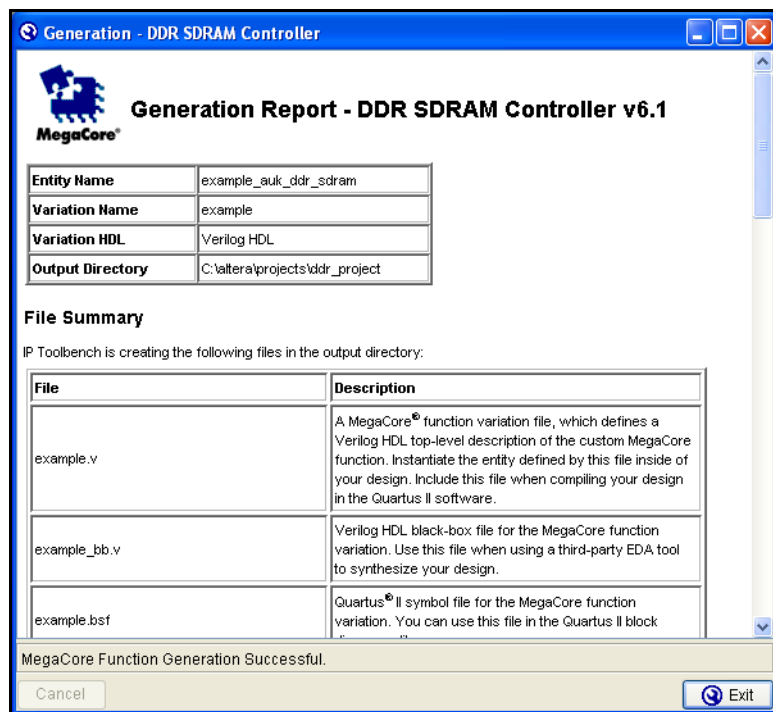




Table 2–1 describes the generated files and other files that may be in your project directory. The names and types of files specified in the the MegaWizard Plug-In Manager report vary based on whether you created your design with VHDL or Verilog HDL.

<b>Table 2–1. Generated Files</b> <i>Note (1)</i>	
<b>Filename</b>	<b>Description</b>
<code>&lt;variation name&gt;.bsf</code>	Quartus II symbol file for the MegaCore function variation. You can use this file in the Quartus II block diagram editor.
<code>&lt;variation name&gt;.html</code>	MegaCore function report file.
<code>&lt;variation name&gt;.vo</code> or <code>.vho</code>	VHDL or Verilog HDL IP functional simulation model.
<code>&lt;variation name&gt;.vhd</code> , or <code>.v</code>	A MegaCore function variation file, which defines a VHDL or Verilog HDL top-level description of the custom MegaCore function. Instantiate the entity defined by this file inside of your design. Include this file when compiling your design in the Quartus II software.
<code>&lt;variation name&gt;_bb.v</code>	Verilog HDL black-box file for the MegaCore function variation. Use this file when using a third-party EDA tool to synthesize your design.
<code>&lt;variation name&gt;_example_driver.vhd</code> , or <code>.v</code>	Example driver.
<code>&lt;variation name&gt;_example_top.vhd</code> , or <code>.v</code>	Example design.

**Notes to Table 2–1:**

(1) `<variation name>` is the variation name.

3. After you review the generation report, click **Exit** to close the MegaWizard Plug-In Manager.
4. Set the `<variation name>_example_top.v` or `.vhd` file to be the project top-level design file.
  - a. Choose

You have finished the walkthrough. Now, simulate the example design (see “[Simulate the Example Design](#)” on page 2–13), edit the PLL(s), and compile (see “[Compile the Example Design](#)” on page 2–15).

## Simulate the Example Design

This section describes the following simulation techniques:

- [Simulate with IP Functional Simulation Models](#)
- [Simulating in Third-Party Simulation Tools Using NativeLink](#)

## Simulate with IP Functional Simulation Models

You can simulate the example design with the the MegaWizard Plug-In Manager-generated IP functional simulation models. the MegaWizard Plug-In Manager generates a VHDL or Verilog HDL testbench for your example design, which is in the **testbench** directory in your project directory.



For more information on the testbench, see “[Example Design](#)” on [page 3–3](#).

You can use the IP functional simulation model with any Altera-supported VHDL or Verilog HDL simulator. The instructions for the ModelSim simulator are different to other simulators.

## Simulating in Third-Party Simulation Tools Using NativeLink

You can perform a simulation in a third-party simulation tool from within the Quartus II software, using NativeLink.



For more information on NativeLink, refer to the *Simulating Altera IP Using NativeLink* chapter in volume 3 of the *Quartus II Handbook*.

To set up simulation in the Quartus II software using NativeLink, follow these steps:

1. Create a custom variation with an IP functional simulation model.
2. Set the generated top-level file *<variation name>\_example\_top* to be the project top-level file.
3. Obtain and copy a memory model to a suitable location, for example, the testbench directory.



Before running the simulation you may also need to edit the testbench to match the chosen memory model.

4. Check that the absolute path to your third-party simulator executable is set. On the Tools menu click **Options** and select **EDA Tools Options**.
5. On the Processing menu, point to **Start** and click **Start Analysis & Elaboration**.

6. On the Assignments menu click **Settings**, expand **EDA Tool Settings** and select **Simulation**. Select a simulator under **Tool Name** and in **NativeLink Settings**, select **Compile Test Bench** and click **Test Benches**.
7. Click **New**.
8. Enter a name for the **Test bench name**.
9. Enter the name of the automatically generated testbench, *<variation name>\_example\_top\_tb*, in **Test bench entity**.
10. Enter the name of the top-level instance in **Instance**.
11. Change **Run for** to 500  $\mu$ s.
12. Add the testbench files. In the **File name** field browse to the location of the memory model and the testbench, click **OK** and click **Add**.
13. Click **OK**.
14. Click **OK**.
15. On the Tools menu point to **EDA Simulation Tool** and click **Run EDA RTL Simulation**.

## Compile the Example Design

To use the Quartus II software to compile the example design and perform post-compilation timing analysis, follow these steps:

1. Enable TimeQuest.
  - a. On the Assignments menu click **Settings**, expand **Timing Analysis Settings**, and select **Use TimeQuest Timing Analyzer**.
  - b. Add the Synopsys design constraints file, *<variation name>\_phy\_ddr\_timing.sdc*, to your project.
2. Add pin I/O standard assignments. Either:
  - a. Run *<variation name>\_pin\_assignments.tcl* (for Stratix III devices)

or

- b. On the Assignments menu click **Pins**. Right-click in the window and click **Create/Import Megafunction**. Select **Import an existing custom megafunction** and navigate to *<variation name>.ppf*.
3. Add the DQ group assignments, to relate the DQ and DQS pin groups together for the Quartus II fitter to place them correctly. (Stratix III devices only), by running *<variation name>\_assign\_dq\_groups.tcl*.
4. Set top-level entity to example project. On the Assignments menu click **Settings**. Click **General**. Under **Top-level entity** select *<variation name>\_example\_top*.
5. On the Processing menu click **Start Compilation**, to compile the design.
6. *Optional*. Run the report timing to get detailed a DDR SDRAM interface timing report. Run *<variation name>\_report\_timing.tcl*:
  - a. On the Tools menu click **Tcl scripts**.

*or*

  - b. On the Tools menu click **TimeQuest Timing Analyzer**. On the Script menu click **Run Tcl Script**.

## Program a Device

After you have compiled the example design, you can perform gate-level simulation (see [“Simulate the Example Design” on page 2–13](#)) or program your targeted Altera device to verify the example design in hardware.

With Altera's free OpenCore Plus evaluation feature, you can evaluate the DDR or DDR2 SDRAM controller MegaCore function before you purchase a license. OpenCore Plus evaluation allows you to generate an IP functional simulation model, and produce a time-limited programming file.



For more information on OpenCore Plus hardware evaluation using the DDR or DDR2 SDRAM controller MegaCore function, see [“OpenCore Plus Evaluation” on page 1–3](#), [“OpenCore Plus Time-Out Behavior” on page 3–3](#), and *Application Note 320: OpenCore Plus Evaluation of Megafunctions*.

## Implement Your Design

To implement your design based on the example design, replace the example driver in the example design with your own logic.

## Set Up Licensing

You need to purchase a license for the MegaCore function only when you are completely satisfied with its functionality and performance, and want to take your design to production.

After you purchase a license for DDR or DDR2 SDRAM controller MegaCore function, you can request a license file from the Altera web site at [www.altera.com/licensing](http://www.altera.com/licensing) and install it on your computer. When you request a license file, Altera emails you a **license.dat** file. If you do not have Internet access, contact your local Altera representative.



### Functional Description

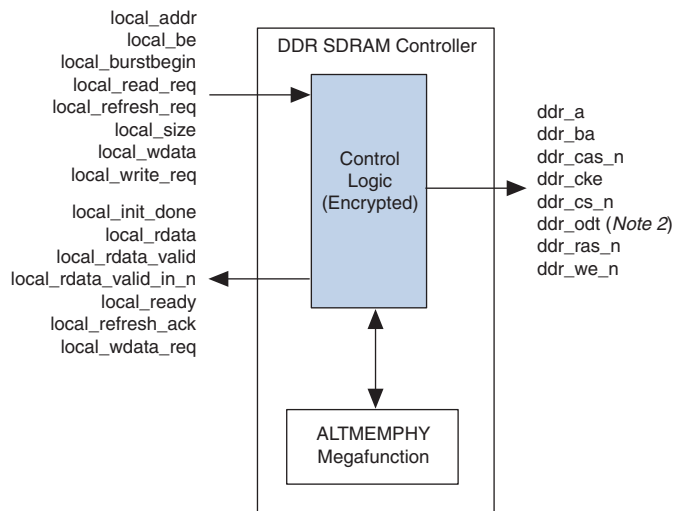
The DDR and DDR2 SDRAM controllers instantiate an encrypted control logic and the ALTMEMPHY megafunction.



For more information on the ALTMEMPHY megafunction, refer to Quartus II Help.

Figure 3–1 shows a block diagram of the DDR & DDR2 SDRAM controller.

**Figure 3–1. DDR & DDR2 SDRAM Controller Block Diagram** *Note (1)*



**Notes to Figure 3–1:**

- (1) You can edit the ddr prefix on the SDRAM interfaces signals.
- (2) DDR2 SDRAM controller only.

### Control Logic

Bus commands control SDRAM devices using combinations of the ddr\_ras\_n, ddr\_cas\_n, and ddr\_we\_n signals. For example, on a clock cycle where all three signals are high, the associated command is a

no operation (NOP). A NOP command is also indicated when the chip select signal is not asserted. Table 3–1 shows the standard SDRAM bus commands.

<b>Table 3–1. Bus Commands</b>				
<b>Command</b>	<b>Acronym</b>	<b>ras_n</b>	<b>cas_n</b>	<b>we_n</b>
No operation	NOP	High	High	High
Active	ACT	Low	High	High
Read	RD	High	Low	High
Write	WR	High	Low	Low
Burst terminate	BT	High	High	Low
Precharge	PCH	Low	High	Low
Auto refresh	ARF	Low	Low	High
Load mode register	LMR	Low	Low	Low

The DDR and DDR2 SDRAM controllers must open SDRAM banks before they access addresses in that bank. The row and bank to be opened are registered at the same time as the active (ACT) command. The DDR and DDR2 SDRAM controllers close the bank and open it again if they need to access a different row. The precharge (PCH) command closes a bank.

The primary commands used to access SDRAM are read (RD) and write (WR). When the WR command is issued, the initial column address and data word is registered. When a RD command is issued, the initial address is registered. The initial data appears on the data bus 2 to 3 clock cycles later (3 to 5 for DDR2 SDRAM). This delay is the column address strobe (CAS) latency and is due to the time required to read the internal DRAM core and register the data on the bus. The CAS latency depends on the speed of the SDRAM and the frequency of the memory clock. In general, the faster the clock, the more cycles of CAS latency are required. After the initial RD or WR command, sequential reads and writes continue until the burst length is reached or a burst terminate (BT) command is issued. DDR and DDR2 SDRAM devices support burst lengths of 2, 4, or 8 data cycles. The auto-refresh command (ARF) is issued periodically to ensure data retention. This function is performed by the DDR or DDR2 SDRAM controller.

The load mode register command (LMR) configures the SDRAM mode register. This register stores the CAS latency, burst length, and burst type.



For more information, refer to the specification of the SDRAM that you are using.



## OpenCore Plus Time-Out Behavior

OpenCore® Plus hardware evaluation can support the following two modes of operation:

- *Untethered*—the design runs for a limited time
- *Tethered*—requires a connection between your board and the host computer. If tethered mode is supported by all megafunctions in a design, the device can operate for a longer time or indefinitely

All megafunctions in a device time out simultaneously when the most restrictive evaluation time is reached. If there is more than one megafunction in a design, a specific megafunction's time-out behavior may be masked by the time-out behavior of the other megafunctions.



For MegaCore functions, the untethered timeout is 1 hour; the tethered timeout value is indefinite.

Your design stops working after the hardware evaluation time expires and the `local_ready` output goes low.



For more information on OpenCore Plus hardware evaluation, see [“OpenCore Plus Evaluation” on page 1–3](#) and *Application Note 320: OpenCore Plus Evaluation of Megafunctions*.

## Example Design

The MegaWizard Plug-In Manager creates an example design that shows you how to instantiate and connect up the DDR or DDR2 SDRAM controller. The example design consists of the DDR or DDR2 SDRAM controller, some driver logic to issue read and write requests to the controller, up to two PLLs to create the necessary clocks and a DLL (Stratix series only). The example design is a working system that can be compiled and used for both static timing checks and board tests.

Figure 3–2 shows the testbench and the example design.

**Figure 3–2. Testbench & Example Design**

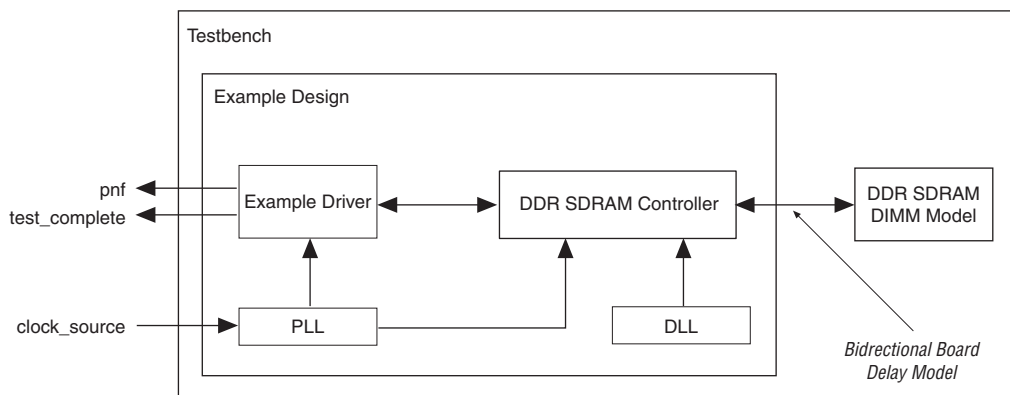


Table 3–2 describes the files that are associated with the example design and the testbench.

<b>Table 3–2. Example Design &amp; Testbench Files</b>	
<b>Filename</b>	<b>Description</b>
<b>&lt;project name&gt;_tb.v or .vhd (1)</b>	Testbench for the example design.
<b>&lt;project name&gt;.v or .vhd (1)</b>	Example design.
<b>ddr_pll_stratix.v or .vhd</b>	Example PLL.
<b>ddr_fb_pll_stratixii.v or .vhd</b>	Optional fed-back PLL (Stratix II devices only).
<b>&lt;variation name&gt;_example_driver.v or .vhd</b>	Example driver.
<b>&lt;variation name&gt;.v or .vhd</b>	Top-level description of the custom MegaCore function.

**Notes to Table 3–2:**

- (1) <project name> is the name of the the MegaWizard Plug-In Manager-generated example design.

The example driver is a self-checking test generator for the DDR or DDR2 SDRAM controller. It uses a state machine to write data patterns to a range of column addresses, within a range of row addresses in all memory banks. It then reads back the data from the same locations, and checks that the data matches. The pass not fail (pnf) output transitions low if any read data fails the comparison. There is also a pnf\_per\_byte output, which shows the comparison on a per byte basis. The

`test_complete` output transitions high for a clock cycle at the end of the write or read sequence. After this transition the test restarts from the beginning.

The data patterns used are generated using an 8-bit LFSR per byte, with each LFSR having a different initialization seed.

The testbench instantiates a DDR or DDR2 SDRAM DIMM model, a reference clock for the PLL, and model for the system board memory trace delays. When `test_complete` is detected high, a test finished message is printed out, which shows whether the test has passed.



Altera does not provide a memory simulation model. You must obtain one from your memory vendor.



For more details on how to run the simulation script, see [“Simulate the Example Design” on page 2-13](#).

## Interfaces & Signals

This section describes the following topics:

- [“Interface Description” on page 3-5](#)
- [“Signals” on page 3-16](#)

### Interface Description

This section describes the following local-side interface requests:

- [“Writes” on page 3-5](#)
- [“Reads” on page 3-7](#)
- [“Read-Write-Read-Write” on page 3-9](#)
- [“Read-Write-Read-Write” on page 3-9](#)
- [“DDR SDRAM Initialization Timing” on page 3-12](#)
- [“DDR2 SDRAM Initialization Timing” on page 3-14](#)



These interface requests are for the native interface. For the Avalon™ Memory-Mapped interface see the *Avalon Memory-Mapped Interface Specification*.

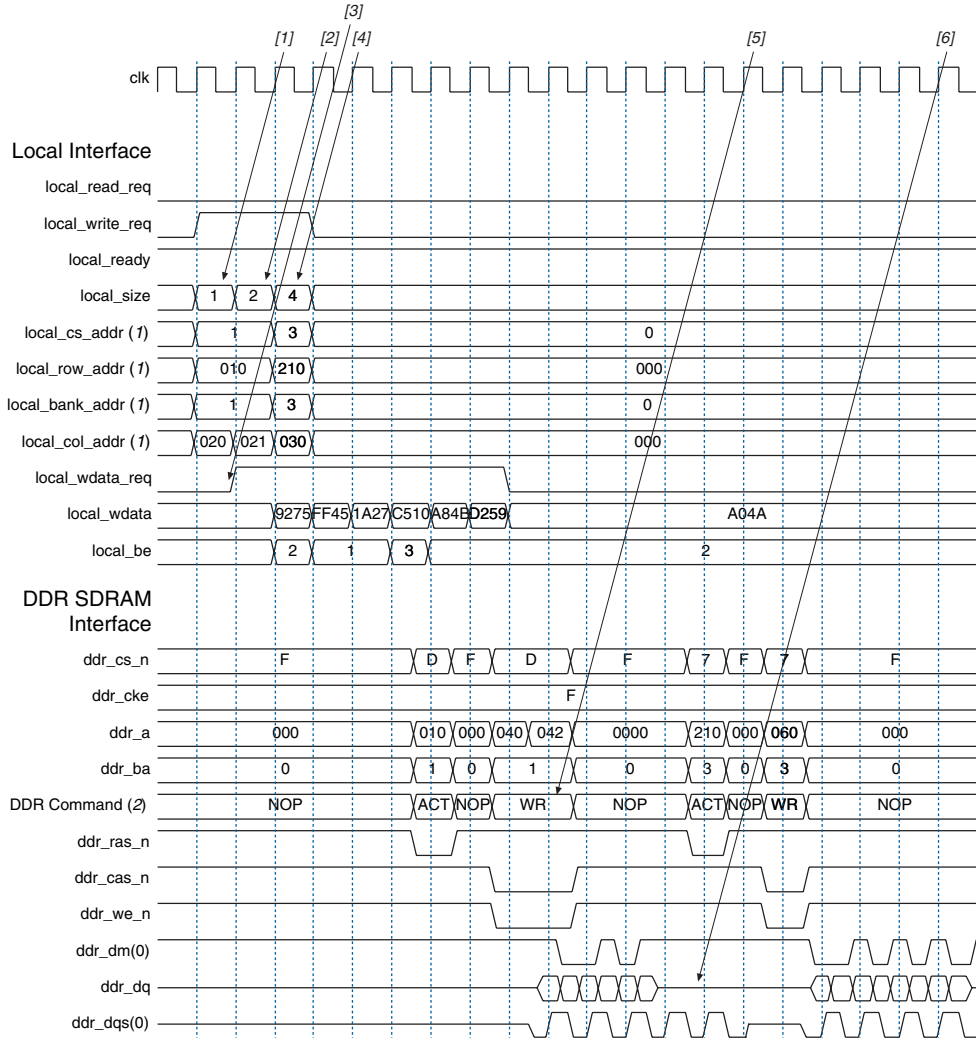
#### *Writes*

[Figure 3-3 on page 3-6](#) shows three write requests of different sizes, the first two to sequential addresses and the third to a new row and bank. The controller allows you to use any burst length up to the maximum burst length set on the memory device. For example, if you select burst length of 8 for your DDR SDRAM memory, the controller allows bursts of length 1, 2, 3, and 4 (2, 4, 6, and 8 on the DDR SDRAM side).



The concept is similar for DDR2 SDRAM although only burst lengths 1 and 2 (2 and 4 on the DDR2 SDRAM side) are available.

**Figure 3–3. Writes**



**Notes to Figure 3–3:**

- (1) The `local_cs_addr`, `local_row_addr`, `local_bank_addr`, and `local_col_addr` signals are a representation of the `local_addr` signal.
- (2) DDR Command shows the command that the command signals (`ddr_ras_n`, `ddr_cas_n` and `ddr_we_n`) are issuing.

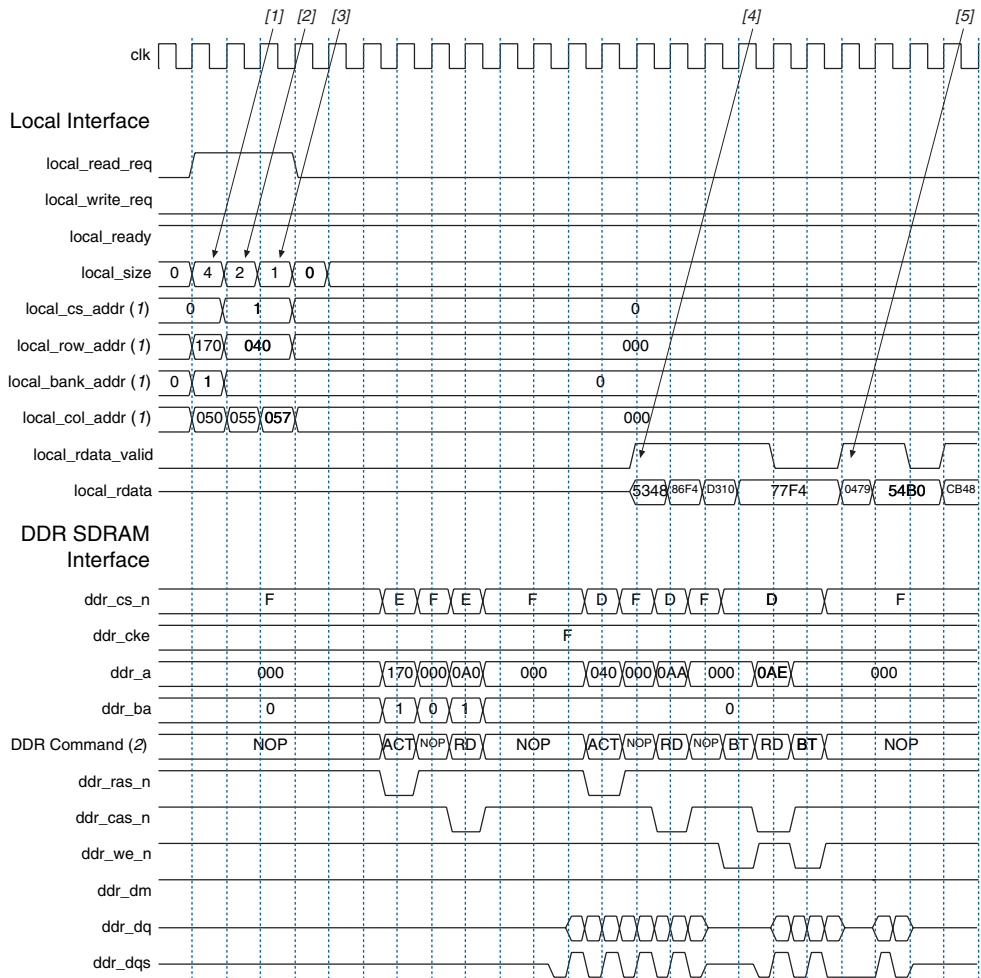
1. The user logic requests the first write, by asserting the `local_write_req` signal, and the size and address for this write. In this example, the request is a burst of length 1 (2 on the DDR SDRAM side) to chip select 1. The `local_ready` signal is asserted, which indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the write request, size, and address signals asserted.
2. The user logic requests a second write to a sequential address, this time of size 2 (4 on the DDR SDRAM side). The `local_ready` signal remains asserted, which indicates that the controller has accepted the request.
3. The controller requests the write data and byte enables for the first write from the user logic. The write data and byte enables must be presented in the clock cycle after the request. In this example, the controller also continues to request write data for the subsequent writes. The user logic must be able to supply the write data for the entire burst when it requests a write.
4. The user logic requests the third write to a different chip select. The controller is able to buffer up to four requests so the `local_ready` signal stays high and the request is accepted.
5. When it has issued the necessary bank activation command, the controller issues the first two write requests sequentially to the memory device.
6. Even though no data is being written to memory, the `ddr_dqs` signal must continue toggling for the entire length of the memory device's burst length (8 in this example).

### *Reads*

Figure 3–4 on page 3–8 shows three read requests of different sizes. The controller allows you to use any burst length up to the maximum burst length set on the memory device. For example, if you select burst length of 8 for your DDR SDRAM memory, the controller allows bursts of length 1, 2, 3, and 4 (2, 4, 6, and 8 on the DDR SDRAM side).



The concept is similar for DDR2 SDRAM although only burst lengths 1 and 2 (2 and 4 on the DDR2 SDRAM side) are available.

**Figure 3–4. Reads****Notes to Figure 3–4:**

- (1) The `local_cs_addr`, `local_row_addr`, `local_bank_addr`, and `local_col_addr` signals are a representation of the `local_addr` signal.
- (2) DDR Command shows the command that the command signals are issuing.

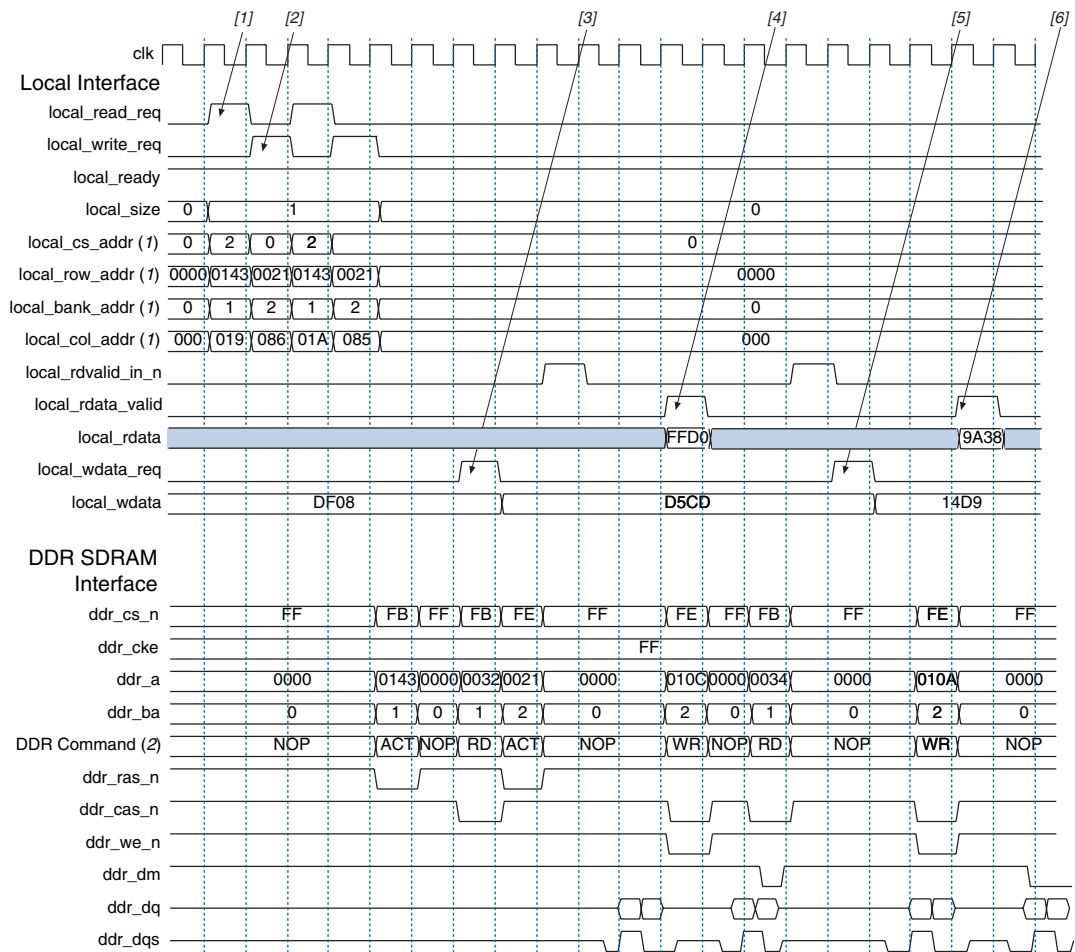
1. The user logic requests the first read by asserting the `local_read_req` signal, and the size and address for this read. In this example, the request is a burst of length 4 (8 on the DDR SDRAM side). The `local_ready` signal is asserted, which

indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the read request, size, and address signals asserted.

2. The user logic requests a second read to a different address, this time of size 2 (4 on the DDR SDRAM side). The `local_ready` signal remains asserted, which indicates that the controller has accepted the request.
3. The user logic requests a third read to a different address, this time of size 1 (2 on the DDR SDRAM side). The `local_ready` signal remains asserted, which indicates that the controller has accepted the request.
4. The controller returns the read data for the first request by asserting the `local_rdata_valid` signal. The exact number of clock cycles between the controller accepting the request and returning the data depends on the number of other requests pending in the controller, the state the memory is in, and the timing requirements of the memory (e.g., the CAS latency).
5. The controller returns the read data for the subsequent read requests.

### *Read-Write-Read-Write*

Figure 3–5 on page 3–10 shows a sequence of interleaved reads and writes.

**Figure 3–5. Read-Write-Read-Write**

**Notes to Figure 3–5:**

- (1) The `local_cs_addr`, `local_row_addr`, `local_bank_addr`, and `local_col_addr` signals are a representation of the `local_addr` signal.
- (2) DDR Command shows the command that the command signals are issuing.

1. The user logic requests a read request by asserting the `local_read_req` signal along with the size and address for that read. Because the `local_ready` signal is high, that request can be considered accepted.
2. The user logic requests a write, a read, and another write request, which are accepted.

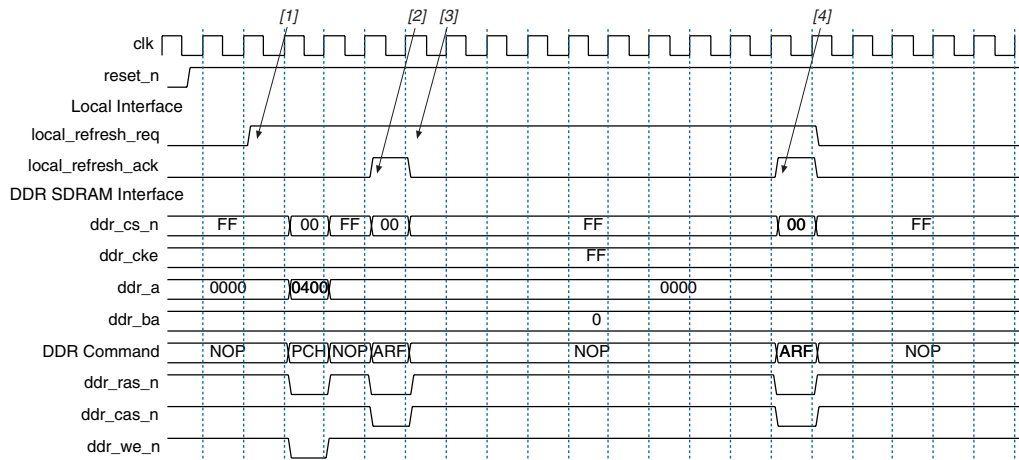


3. The controller asserts the write data request signal to ask the user logic to present valid write data and byte enables on the next clock edge.
4. The read data from the first read request is returned and marked as valid by the read data valid signal.
5. The controller again asserts the write data request for the second write request.
6. The read data from the second read request is returned.

### User Refresh Control

Figure 3–6 shows the user refresh control interface. This feature allows you to control when the controller issues refreshes to the memory. This feature allows better control of worst case latency and allows refreshes to be issued in bursts to take advantage of idle periods.

**Figure 3–6. User Refresh Control**



**Note to Figure 3–6:**

- (1) DDR Command shows the command that the command signals are issuing.

1. The user logic asserts the refresh request signal to indicate to the controller that it should perform a refresh. The state of the read and write requests signal does not matter as the controller gives priority to the refresh request (although it completes any currently active reads or writes).

2. The controller asserts the refresh acknowledge signal to indicate that it has issued a refresh. This signal is still available even if the user refresh control option is not switched on, allowing the user logic to keep track of when the controller is issuing refreshes.
3. The user logic keeps the refresh request signal asserted to indicate that it wishes to perform another refresh request.

The controller again asserts the refresh acknowledge signal to indicate that it has issued a refresh. At this point the user logic deasserts the refresh request signal and the controller continues with the reads and writes in its buffers.

### *DDR SDRAM Initialization Timing*

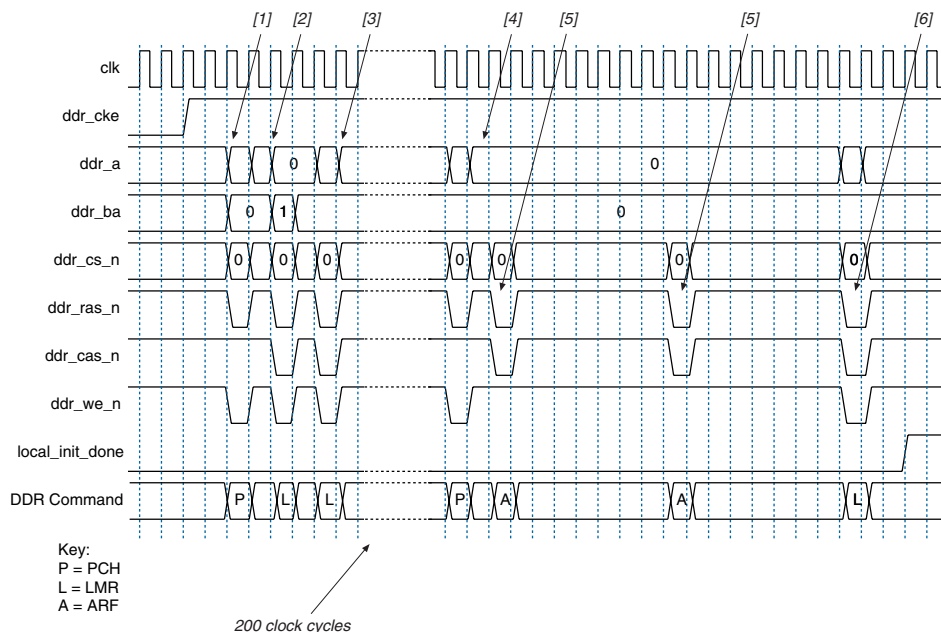


DDR SDRAM and DDR2 SDRAM initialization timing is different. For DDR2 SDRAM initialization timing, see [“DDR2 SDRAM Initialization Timing” on page 3–14](#).

The DDR SDRAM controller initializes the SDRAM devices by issuing the following memory command sequence:

- NOP (for 200  $\mu$ s, programmable)
- PCH
- Extended LMR (ELMR)
- LMR
- NOP (for 200 clock cycles, fixed)
- PCH
- ARF
- ARF
- LMR

[Figure 3–7 on page 3–13](#) shows a typical initialization timing sequence, which is described below. The length of time between the reset and the first PCH command should be 200  $\mu$ s. This time can be reduced for simulation testing by setting the start-up timer parameter in the MegaWizard Plug-In Manager.

**Figure 3–7. DDR SDRAM Device Initialization Timing**

1. A PCH command is sent to all banks by setting the precharge pin, the address bit a [10], or a [8] high.
2. An ELMR command is issued to enable the internal delay-locked loop (DLL) in the memory devices. An ELMR command is an LMR command with the bank address bits set to address the extended mode register.
3. An LMR command sets the operating parameters of the memory such as CAS latency and burst length. This LMR command is also used to reset the internal memory device DLL. The DDR SDRAM controller allows 200 clock cycles to elapse after a DLL reset and before it issues the next command to the memory.
4. A further PCH command places all the banks in their idle state.
5. Two ARF commands must follow the PCH command.
6. The final LMR command programs the operating parameters without resetting the DLL.

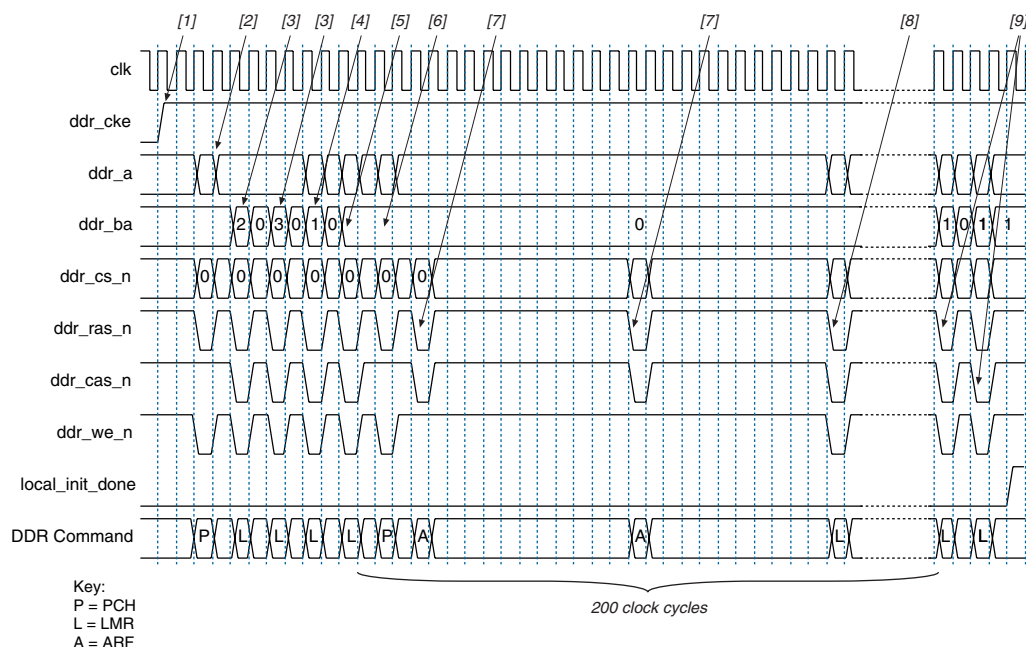
The DDR SDRAM controller asserts the `local_init_done` signal, which shows that it has initialized the memory devices.

### *DDR2 SDRAM Initialization Timing*

The DDR2 SDRAM controller initializes the memory devices by issuing the following command sequence:

- NOP (for 200  $\mu$ s, programmable)
- PCH
- ELMR, register 2
- ELMR, register 3
- ELMR, register 1
- LMR
- PCH
- ARF
- ARF
- LMR
- ELMR, register 1
- ELMR, register 1

Figure 3–8 on page 3–15 shows a typical DDR2 SDRAM initialization timing sequence, which is described below. The length of time between the reset and the clock enable signal going high should be 200  $\mu$ s. This time can be reduced for simulation testing by setting the start-up timer parameter in the MegaWizard Plug-In Manager.

**Figure 3–8. DDR2 SDRAM Device Initialization Timing**

1. The clock enable signal (CKE) is asserted 200  $\mu$ s after coming out of reset.
2. The controller then waits 400 ns and then issues the first PCH command by setting the precharge pin, the address bit a[10] or a[8] high. The 400 ns is calculated by taking the number of clock cycles calculated by the wizard for the 200  $\mu$ s delay and dividing this by 500. If a small initialization time is selected for simulation purposes, this delay is always at least 1 clock cycle.
3. Two ELMR commands are issued to load extend mode registers 2 and 3 with zeros.
4. An ELMR command is issued to extend mode register 1 to enable the internal DLL in the memory devices.
5. An LMR command is issued to set the operating parameters of the memory such as CAS latency and burst length. This LMR command is also used to reset the internal memory device DLL.
6. A further PCH command places all the banks in their idle state.

7. Two ARF commands must follow the PCH command.
8. A final LMR command is issued to program the operating parameters without resetting the DLL.
9. 200 clock cycles after step 5, two ELMR commands are issued to set the memory device off-chip driver (OCD) impedance to the default setting.

The DDR2 SDRAM controller asserts the `local_init_done` signal, which shows that it has initialized the memory devices.

## Signals

Table 3–3 shows the DDR and DDR2 SDRAM controller local interface signals.

<b>Table 3–3. Local Interface Signals (Part 1 of 3)</b>		
<b>Signal Name</b>	<b>Direction</b>	<b>Description</b>
<code>local_addr[]</code>	Input	<p>Memory address at which the burst should start. The width of this bus is sized using the following equation:</p> <p>For one chip select: width = bank bits + row bits + column bits – 1</p> <p>For multiple chip selects: width = chip bits + bank bits + row bits + column bits – 1</p> <p>The least significant bit (LSB) of the column address on the memory side is ignored, because the local data width is twice that of the memory data bus width.</p> <p>The order of the address bits is set in the clear text part of the MegaCore function (<code>auk_dds_sdrn.vhd</code>). The order is chips, bank, row, column, but you can change it if required.</p>
<code>local_be[]</code>	Input	Byte enable signal, which you use to mask off individual bytes during writes.
<code>local_burstbegin</code>	Input	Avalon burst begin strobe, which indicates the beginning of an Avalon burst. This signal is only available when the local interface is an Avalon interface and the memory burst length is greater than 2.
<code>local_read_req</code>	Input	Read request signal.

**Table 3–3. Local Interface Signals (Part 2 of 3)**

Signal Name	Direction	Description
<code>local_refresh_req</code>	Input	User controlled refresh request. If User Controlled Refresh is turned on, <code>local_refresh_req</code> becomes available and you are responsible for issuing sufficient refresh requests to meet the memory requirements. This option allows complete control over when refreshes are issued to the memory including ganging together multiple refresh commands. Refresh requests take priority over read and write requests unless they are already being processed.
<code>local_size[]</code>	Input	<p>The burst size of the requested access, which is encoded as a binary number. The controller supports maximum local burst lengths of 1, 2, or 4, for DDR SDRAM; and 2 for DDR2 SDRAM.</p> <p>You may request any size up to the maximum burst length, so for example if you chose a memory burst length of 8, the local burst size is 4 and you may request local bursts of length 1, 2, 3 or 4. Similarly, if you chose a memory burst length of 4, the local burst length is 2 and you may request local bursts of length 1 or 2.</p> <p>If you chose a memory burst length of 2 (local burst length of 1), the <code>local_size[]</code> port is tied to 1 and is not visible on the controller interface. For all other memory burst lengths, <code>local_size</code> is available.</p>
<code>local_wdata[]</code>	Input	Write data bus. The width of <code>local_wdata</code> is twice that of the memory data bus.
<code>local_write_req</code>	Input	Write request signal.
<code>local_init_done</code>	Output	Memory initialization complete signal, which is asserted once the controller has completed its initialization of the memory. Read and write requests are still accepted before <code>local_init_done</code> is asserted, however they are not issued to the memory until it is safe to do so.
<code>local_rdata[]</code>	Output	Read data bus. The width of <code>local_rdata</code> is twice that of the memory data bus.
<code>local_rdata_valid</code>	Output	Read data valid signal. The <code>local_rdata_valid</code> signal indicates that valid data is present on the read data bus. The timing of <code>local_rdata_valid</code> is automatically adjusted to cope with your choice of resynchronization and pipelining options.
<code>local_rdvalid_in_n</code>	Output	An early version of the read data valid signal which appears three cycles before it. Not present in Avalon mode.
<code>local_ready</code>	Output	The <code>local_ready</code> signal indicates that the DDR or DDR2 SDRAM controller is ready to accept request signals. If <code>local_ready</code> is asserted in the clock cycle that a read or write request is asserted, that request has been accepted. The <code>local_ready</code> signal is deasserted to indicate that the DDR or DDR2 SDRAM controller cannot accept any more requests.

**Table 3–3. Local Interface Signals (Part 3 of 3)**

Signal Name	Direction	Description
local_refresh_ack	Output	Refresh request acknowledge, which is asserted for one clock cycle every time a refresh is issued. Even if the User Controlled Refresh option is not selected, local_refresh_ack still indicates to the local interface that the controller has just issued a refresh command.
local_wdata_req	Output	Write data request signal, which indicates to the local interface that it should present valid write data on the next clock edge. Not present in Avalon mode.

Table 3–4 shows the DDR and DDR2 SDRAM interface signals.

**Table 3–4. DDR & DDR2 SDRAM Interface Signals** *Note (1)*

Signal Name	Direction	Description
ddr_dq[]	Bidirectional	Memory data bus. This bus is half the width of the local read and write data busses.
ddr_dqs[]	Bidirectional	Memory data strobe signal, which writes data into the DDR or DDR2 SDRAM and captures read data into the Altera device.
clk_to_sdram	Output	Clock for the memory device.
clk_to_sdram_n	Output	Inverted clock for the memory device.
ddr_a[]	Output	Memory address bus.
ddr_ba[]	Output	Memory bank address bus.
ddr_cas_n	Output	Memory column address strobe signal.
ddr_cke[]	Output	Memory clock enable signals.
ddr_cs_n[]	Output	Memory chip select signals.
ddr_dm[]	Output	Memory data mask signal, which masks individual bytes during writes.
ddr_odt	Output	Memory on-die termination control signal (DDR2 SDRAM only).
ddr_ras_n	Output	Memory row address strobe signal.
ddr_we_n	Output	Memory write enable signal.

**Note to Table 3–4:**

- (1) You can change the ddr\_ signal name prefix in the MegaWizard Plug-In Manager.

## Parameters

The parameters can be set only in the MegaWizard Plug-In Manager (see “DDR & DDR2 SDRAM High-Performance Controller Walkthrough” on page 2–2).



Table 3–5 shows the controller settings.

<b>Table 3–5. Controller Settings</b>		
<b>Parameter</b>	<b>Range</b>	<b>Description</b>
Local Interface Settings	Enable user controlled refresh	Turn on for to control the refreshes see “ <a href="#">User Refresh Control</a> ” on <a href="#">page 3–11</a> .
Local Interface Protocol	Native or Avalon Memory-Mapped	Specifies the local side interface between the user logic and the memory controller.

## MegaCore Verification

MegaCore verification involves simulation testing and hardware testing.

### Simulation Testing

Altera has carried out extensive random, directed tests with functional test coverage using industry-standard Denali models to ensure the functionality of the DDR and DDR2 SDRAM controller. In addition, Altera has carried out a wide variety of gate-level tests of the DDR and DDR2 SDRAM controllers to verify the post-compilation functionality of the controllers.

### Hardware Testing

Table 3–6 shows the Altera development boards on which Altera hardware tested the DDR and DDR2 SDRAM controllers.

<b>Table 3–6. Altera Development Boards</b>		
<b>Development Board</b>	<b>Altera Device</b>	<b>Memory Device</b>
Stratix II High-Speed IO Development Board	EP2S60F1020C3	Micron DDR2-533 DIMM (MT8HTF3272AG-53EB3ES)
Stratix II PCI Development Board	EP2S60F1020C3	Infineon DDR400 SO-DIMM (HYS64D32020GDL-5-B)
Stratix PCI Development Board	EP1S25F1020C5	Infineon DDR400 SO-DIMM (HYS64D32020GDL-5-B)
Stratix PCI Development Board, Professional Edition	EP1S60F1020C6	Micron DDR333 SO-DIMM (MT8VDDT3264HG-335C2)
Internal Stratix Memory Test Board	EP1S25F780C5	Micron DDR400 DIMM (MT16VDDT3264AG-40BB5)

