Implementing Multiple Legacy DDR/DDR2 SDRAM Controller Interfaces

July 2007, v2.0 Application Note 392

Introduction

This application note details the steps for designing multiple legacy DDR2 controllers into a single FPGA. After reading this application note you should be able to understand important design considerations such as PLL and DLL resource allocation, timing analysis, and debugging.

If you want to implement ALTMEMPHY based multiple controllers, refer to *AN 462: Implementing Multiple Memory Interfaces Using the ALTMEMPHY Megafunction*, as these will not be discussed here.

The multiple controller design flow is explained by showing an example of a two controller design and a three controller design using the DDR/DDR2 SDRAM controllers.

- Two independent controllers (Figure 1)
  - One on the top and the second on the bottom of the device
  - No resource sharing among the controllers
- Three controllers (Figure 2)
  - Two on the top and another one on the bottom.
  - PLL and DLL resource sharing for the two interfaces on the top of the device

Design Details

This section explains the design specifications, such as memory frequency, memory interface width, and the number of PLLs in the controllers. These details are important to know up front, as they help make it possible to achieve maximum performance by using the correct device. The specifications are as follows:

- Target device: EP2S90F1020C3
- DDR2 Controller version: Quartus® II 6.1 and higher
- Quartus II software: Version 6.1 and higher
- Controller interface frequency: 266.67 MHz for all the controllers in the design
- Interface width: 32 bits for each controller
- Number of PLLs: 2 (System PLL and feedback PLL) for each controller in both designs.)

Figure 1 shows the two controller implementation. In this design, each controller has its own PLL and DLL resources.

Figure 1. Two-Controller Implementation

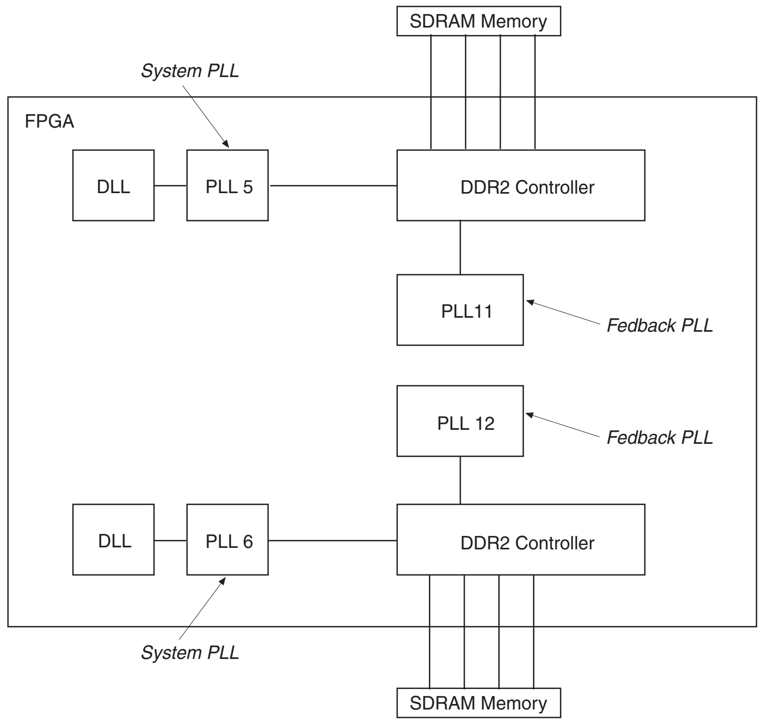
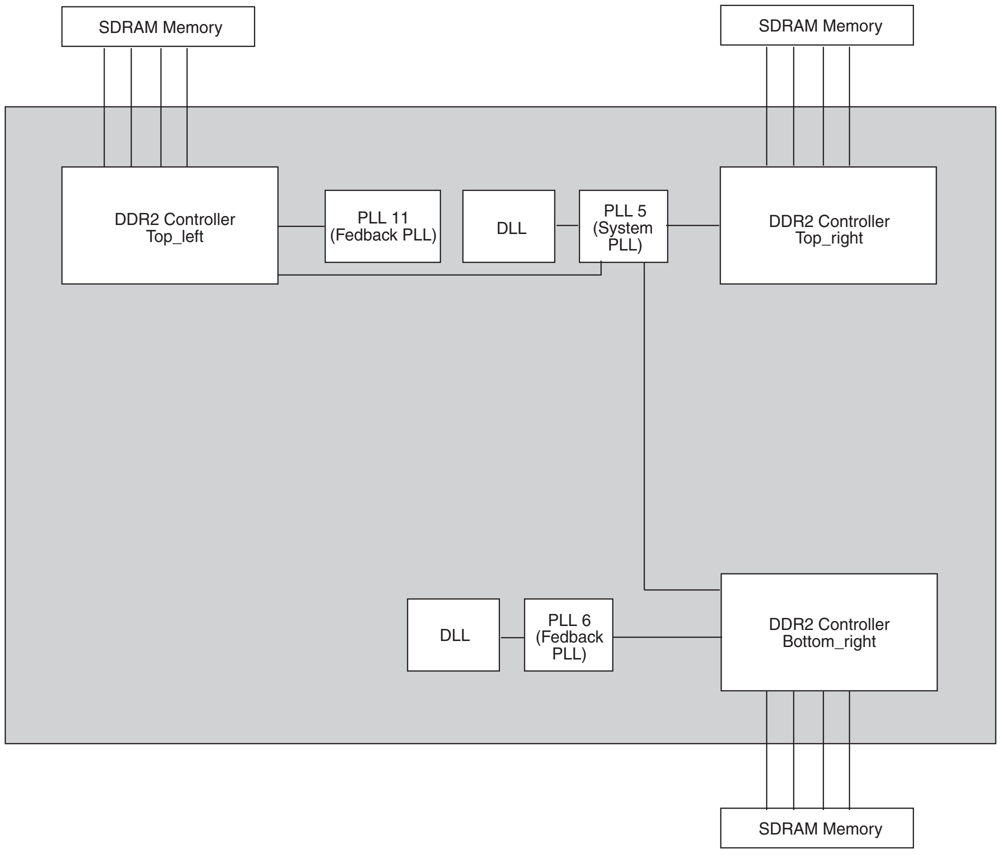


Figure 2 shows the three-controller implementation, in which DLL and PLL resources are shared between two of the controllers.

☞ Refer to Figure 4 for details on which clock resources are shared among the three controllers.

Figure 2. Three-Controller Implementation



Design Considerations

It is important that the designer knows which resources in an FPGA need to be shared between the custom logic and the controllers. The following design considerations apply to both two- and three-controller designs:

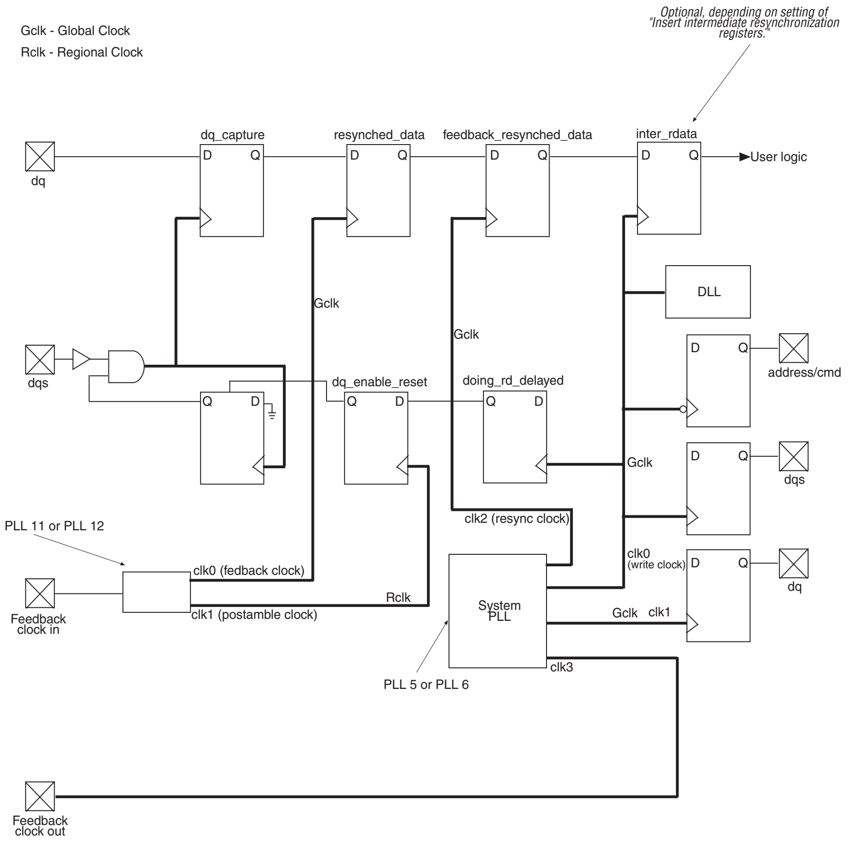
1. One or more clock pairs (`clk_to_sdram[]` and `clk_to_sdram_n[]`) from the FPGA to memory is available for each interface.

2. One system PLL and one feedback PLL are available for each controller. The system PLL generates the system clock and the feedback PLL generates the feedback clock in each controller. Figure 3 shows the configuration Altera® recommends for closing timing on legacy-based memory interfaces greater than 200 MHz.

Refer to the section *Appendix A, Resynchronization* in the *DDR and DDR2 SDRAM Controller Compiler User Guide* to learn more about the functionality of the feedback and system PLL.

Figure 3 shows a two-PLL read datapath implementation, including the PLLs configured as system and feedback PLLs.

Figure 3. A DDR2 SDRAM Controller Using Both a Feedback and a System PLL



DLL Resource Considerations

The Stratix® II and Stratix II GX series of FPGA devices support one DLL on the top and one DLL on the bottom of the FPGA. This requires that memory controllers on the same side of the device operate at the same frequency, because all the controllers must share the output from the common DLL.

PLL Resource Considerations

PLL resources are critical because only a fixed number of PLL output taps are available. You should be careful when choosing global and regional clock resources, as they are limited in number and vary depending upon the device. This section discusses what type of resource to use for different clocks.

To learn more about the PLL resources, refer to the *Enhanced PLLs* and *Fast PLLs* sections of the *PLLs in Stratix II and Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook*.

In a two-controller design, because the two controllers are independent, the PLLs are instantiated by the IP tool bench and require no modifications.

If both the memory interfaces are at the same frequency, then the PLL resources can be shared.

If the data has to cross over boundaries that are clocked by clocks coming out of two separate PLLs, proper data synchronization care should be exercised.

In a three-controller design, the minimum number of PLL taps required to drive all three controllers is 11. Table 1 describes the clocks required for the three-controller example design.

Table 1. Clocks Required for Three-Controller Example Design

Clock Name	Description	# of Clocks
System clock	A single system clock drives all 3 controllers.	1
Write clock	A single write clock drives all 3 controllers.	1
Resynchronization clock	There is one resynchronization clock for each controller. *(1)*	3
Feedback resynchronization clocks	There is one feedback resynchronization clock for each controller. *(1)*	3
Postamble clocks	There is one postamble clock for each controller. *(1)*	3
Total		11

Note to Table 1:

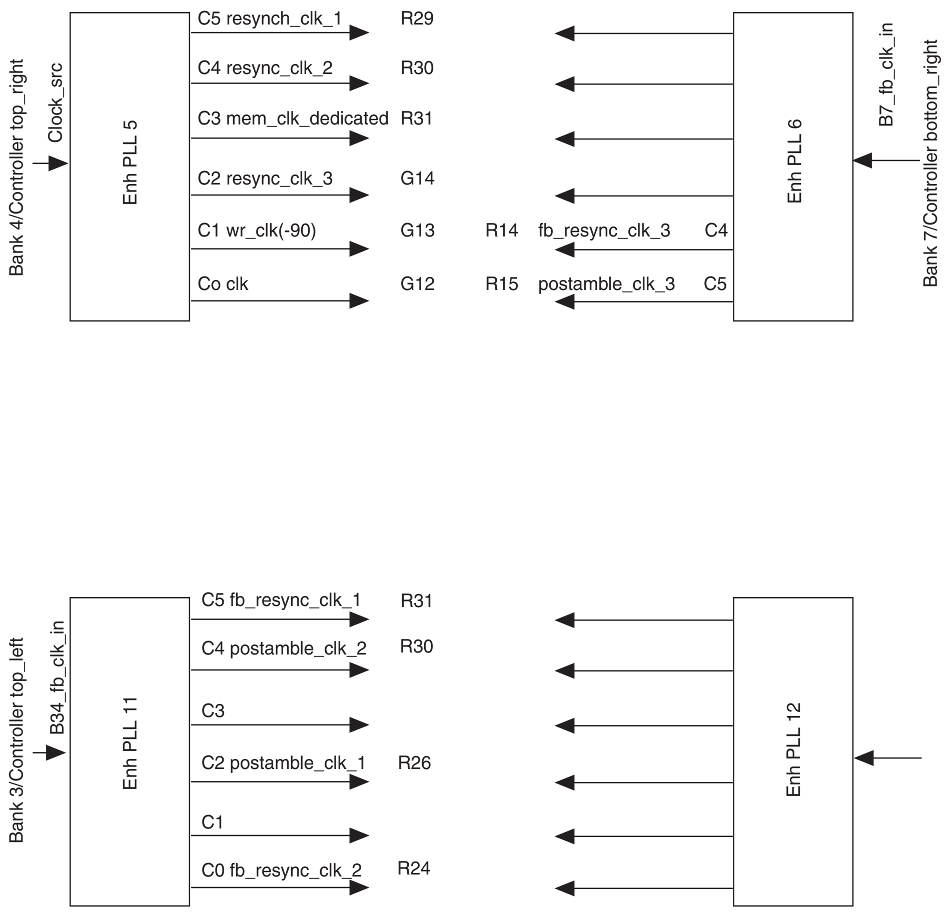
(1) If two or more controllers share a resynchronization/feedback/postamble clock, you must place matching constraints on the board trace delay and the board layout for the external memories. To overcome such constraints, use three separate resynchronization/feedback/postamble clocks. The 11 PLL taps are chosen from three enhanced PLLs, as shown in Figure 4.

The DLL reference clock for I/O banks 3 and 4 can only be driven by enhanced PLL5 or by the dedicated clock input pins `CLK[4-7]` or `CLK[12-15]` on the appropriate side of the device. The DLL reference clock for I/O banks 7 and 8 can only be driven by enhanced PLL6 or by the dedicated clock input pins on the appropriate side of the device. For example, if the PLL input reference clock frequency is 100 MHz, and the required output frequency is 266.67 MHz, the multiplication factor will be 8/3. In this case, the DLL reference clock, which must have a frequency of 266 MHz, cannot be driven by any of the dedicated clock input pins and therefore must be driven by enhanced PLL5 or enhanced PLL6.

Do not use any feedback clocks to drive the DLL reference because feedback clocks generally have a higher jitter due to the signal going off-chip and then back on-chip. The system PLL must be PLL5 for the controllers at the top and PLL6 for the controllers at the bottom.

Figure 4 shows the PLL resource usage in a three-controller design. The figure also shows which PLL tap is used as a global/regional resource.

Figure 4. PLL Resource Allocation



Note to Figure 4:

(1) `mem_clk-dedicated` is optional in FPGA designs but is required to drive the `clk_to_sdram` outputs in HardCopy® II applications.

Using three enhanced PLLs, as shown in Figure 4, requires the fewest number of global routing resources. Table 2 describes the tradeoffs you must consider before using fast PLLs or fewer enhanced PLLs in your three-controller design.

Table 2. Tradeoffs Between Enhanced PLLs and Fast PLLs

PLL Possibilities	Comments
2 enhanced PLLs — Both on the top	Only 4 chip-wide global signals can be driven from any side of the device. However, the system clock, write clock, resynchronization clock, feedback resynchronization clock, and postamble clock for the controller on the bottom of the chip all require global clocks, which is 1 more than is available. This could still be accomplished by sharing one of the resynchronization, feedback resynchronization, or postamble clocks with another interface, but that would cause board layout restrictions. All resynchronization, feedback resynchronization, and postamble clocks for the top DDR interfaces must be dual-regional.
2 enhanced PLLs — 1 on the top and 1 on the bottom	If the system PLL is placed on the top with 2 DDR interfaces, the system clock, write clock, and resynchronization clock for the bottom interface must be driven onto global networks, leaving 1 global network free on the top. The feedback PLL would require all 4 bottom global networks to reach the DDR interfaces at the top. If the system PLL is placed on the bottom with the single DDR interface, the system clock, write clock, and 2 resynchronization clocks must be made global from the bottom of the chip. The feedback PLL at the top would only need to drive 2 signals globally from the top. Therefore, putting the feedback PLL on the same side of the chip as the 2 DDR interfaces is a more desirable configuration.
2 enhanced PLLs and 1 fast PLL	The purpose of the feedback PLL is primarily to compensate for the variation of the I/O buffer input and output delays. If you use a fast PLL for the feedback PLL, you are using I/Os on the side of the chip and not on the top or bottom. Since those pins have different characteristics, they will not track the variations as well. Fast PLLs have only 4 output taps, so you need more than 1 of them to support 3 DDR interfaces. Another concern is that a fast PLL requires a global network to drive signals to the opposite side of the chip (that is, left to right or right to left), so you may need to use a global signal to drive a signal that only needs to go to the top or to the bottom. Depending on the number of memory interfaces, using a fast PLL as the system PLL may still be feasible.

Table 3 shows that the system clock (`clk`), write clock (`wr_clk`) and the resynchronization clock (`resync_clk_3`) are made global as represented by `Gclk`. This is because the system clock and the write clock have to drive all three controllers on both sides of the chip, and the resynchronization clock (`resync_clk_3`) is made global as the signal has to traverse from the top of the chip to the bottom of the chip to drive the resynchronization registers of the bottom right controller.

Table 3. Clock Resource Types Note (1)

Signals	Enhanced PLL #	Resource Type	Comments
`clk`	11	Global	The clock drives the 3 controllers that are placed in three quadrants.
`write_clk`	11	Global	The clock drives the 3 controllers that are placed in three quadrants.
`dedicated_resynch_or_capture_clk_bottom_right`	11	Global	The signal has to traverse across the quadrants.
`mem_clk_dedicated`	11	None	Only needed to drive dedicated clock output pins. Not required.
`dedicated_resynch_or_capture_clk_top_right`	11	Regional	—
`dedicated_resynch_or_capture_clk_top_left`	11	Regional	—
`fedback_resynch_clk_top_left`	5	Regional	—
`dedicated_postamble_clk_top_left`	5	Regional	—
`dedicated_postamble_clk_top_right`	5	Regional	—
`fedback_resynch_clk_top_right`	5	Regional	—
`fedback_resynch_clk_bottom_right`	6	Regional	—
`dedicated_postamble_clk_bottom_right`	6	Regional	—

Note to Table 3:

(1) These settings for the regional and global clocks work when each of the controllers has a 32-bit or narrower DQ interface. If the DQ interface exceeds 32 or 40 bits, it spans the entire top or bottom side of the chip. In that case, dual-regional clocks must be used, because the interface is spread across two quadrants. The PLL routing resources vary for each design, and the example design illustrates one case.

Instantiating DDR/DDR2 SDRAM Memory Controllers

This section explains how to generate controllers for the design. It is important to go through the section thoroughly, as it explains the required settings for achieving the proper timing with minimum iterations.

After creating the Quartus II project, generate DDR/DDR2 SDRAM controllers using the MegaWizard® Plug-In Manager.

☞ You must invoke the MegaWizard Plug-In Manager each time you generate a new controller under the same project.

Refer to the *DDR and DDR2 SDRAM Controller Compiler User Guide* to learn more about parameterizing and generating the SDRAM controller using the MegaWizard Plug-In Manager.

When generating cores using the MegaWizard Plug-In Manager, four pages are critical:

- MegaWizard Plug-In Manager Page 2a
- IP Toolbench - Step 1: Parameterize - Controller Page
- IP Toolbench - Step 1: Parameterize - Project Settings Page
- IP Toolbench - Step 1: Parameterize - Manual Timings Page

MegaWizard Plug-In Manager Page 2a

Generate the controllers in separate directories. For example, if the design has two controllers, one at the top and one at the bottom, generate the DDR2 controller megafunctions at <*quartus_project_dir*>**/top_ddr2.v** and <*quartus_project_dir*>**/bottom_ddr2.v**. This helps you avoid the clutter of having all the files in one directory.

When the controllers are generated in separate directories, the PLLs are instantiated with the same names (`ddr_pll_stratixii` and `ddr_fb_pll_stratixii`) for both the controllers. Change the names of the PLL instantiations so that they are not common to the two controllers in the design. Also, when you regenerate the controller for any reasons, the PLL files (**ddr_pll_stratixii.v** and **ddr_fb_pll_stratixii.v**) are overwritten and you must repeat the name change process.

When you generate the controllers, the MegaWizard Plug-In Manager generates two PLLs: the system PLL and the feedback PLL. The PLLs are instantiated in the file **multiple_ddr2_top.v** under the directories **top_right**, **top_left** and **bottom_right**. For the three-controller design, these PLLs are not used in the top-level design and three separate PLLs are generated using the MegaWizard Plug-In Manager to suit the design requirements. Generate the PLLs under the directories **pll_1_top**, **pll_2_top**, and **pll_bottom**. Refer to the top-level design file of the three-controller design (**multiple_ddr2_top.v**) to see the PLL instantiations along with their output connections.

Controller Page

a. Turn on **DQS mode**.

b. Turn on **Use fedback clock**.

When this option is enabled, the registers that capture data from the DQ pins during reads are clocked by a delayed version of DQS. Otherwise, a PLL-generated clock captures the data (Stratix series only). DQS mode provides higher performance than non-DQS mode. Non-DQS mode allows greater flexibility for placing pins, because it allows DQ pins to be placed in the side banks. Only the top and bottom banks support the circuitry required to delay the DQS signals to capture the read data.

For more details about DQS and non-DQS mode, refer to *AN 328: Interfacing DDR2 SDRAM with Stratix II Devices*.

c. Turn on **Insert pipeline registers on address and command outputs**.

This register helps to achieve the required performance at frequencies of 200 MHz. When this option is enabled, the wizard inserts a pipeline register stage between the memory controller and the command and address outputs. When this option is turned on, an extra cycle of latency is added between the time at which the `local_read_req` or `local_write_req` signal is asserted at the local interface and the time address/command appears at the memory interface.

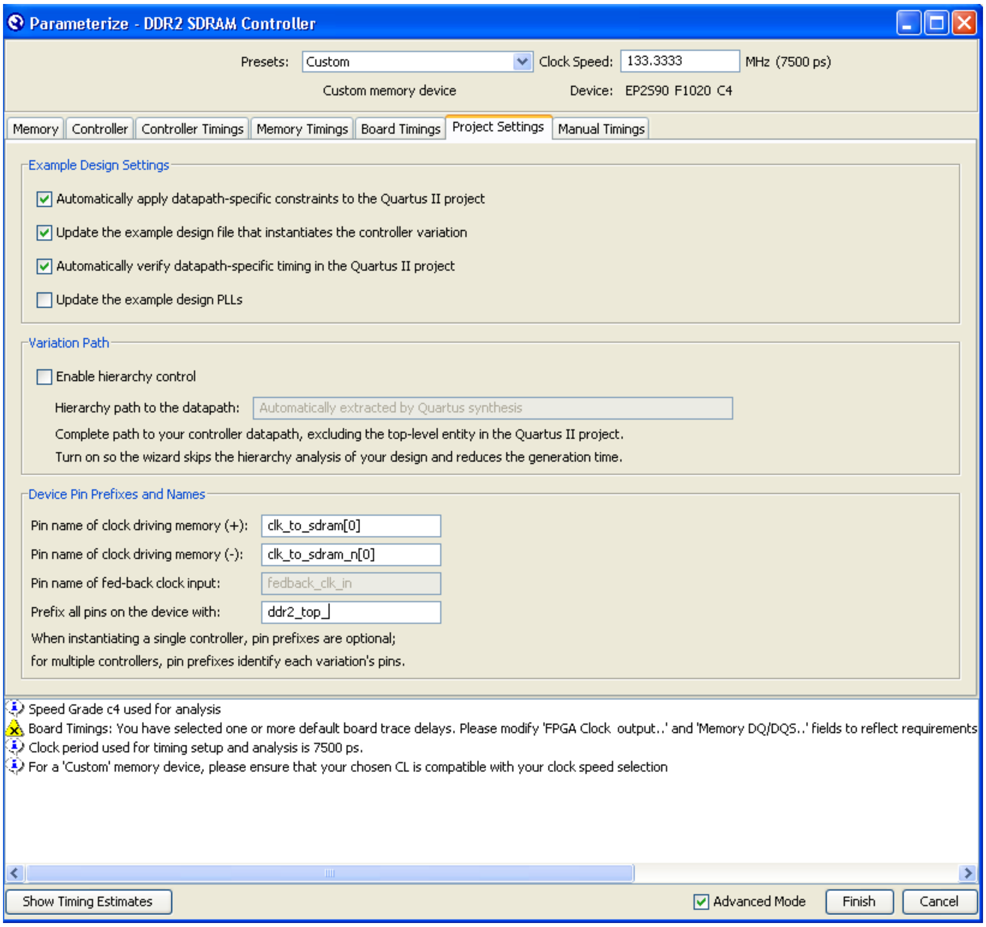
d. Turn on **Clock address/command output registers on the negative edge**.

This option allows you to adjust the output timing of the address and command signals to meet the setup and hold requirements of the memory device. However, you must perform your own timing analysis of address/command timing. Generally, turn on this option, except for Stratix II designs operating at 200 MHz or higher.

Project Settings Page

Prefix all pins on the device with appropriate names while generating the respective controller. By doing this, the IP Toolbench prefixes the top level pin names of the controller with the name given, as shown in Figure 5.

Figure 5. Project Settings Page



Parameterize - DDR2 SDRAM Controller

Presets: Custom — Clock Speed: 133.3333 MHz (7500 ps)

Custom memory device — Device: EP2S90 F1020 C4

Example Design Settings

- [✓] Automatically apply datapath-specific constraints to the Quartus II project
- [✓] Update the example design file that instantiates the controller variation
- [✓] Automatically verify datapath-specific timing in the Quartus II project
- [ ] Update the example design PLLs

Variation Path

- [ ] Enable hierarchy control

Hierarchy path to the datapath: Automatically extracted by Quartus synthesis

Complete path to your controller datapath, excluding the top-level entity in the Quartus II project.
Turn on so the wizard skips the hierarchy analysis of your design and reduces the generation time.

Device Pin Prefixes and Names

Pin name of clock driving memory (+):	clk_to_sdram[0]
Pin name of clock driving memory (-):	clk_to_sdram_n[0]
Pin name of fed-back clock input:	fedback_clk_in
Prefix all pins on the device with:	ddr2_top_

When instantiating a single controller, pin prefixes are optional;
for multiple controllers, pin prefixes identify each variation's pins.

```
Speed Grade c4 used for analysis
Board Timings: You have selected one or more default board trace delays. Please modify 'FPGA Clock output..' and 'Memory DQ/DQS..' fields to reflect requirements
Clock period used for timing setup and analysis is 7500 ps.
For a 'Custom' memory device, please ensure that your chosen CL is compatible with your clock speed selection
```

[✓] Advanced Mode

For example in the case of a two-controller design:

a. `ddr2_bottom_`

b. `ddr2_top_`

Turn off **Update the example design PLLs**, if PLL phases have to be changed later and edited by using the PLL MegaWizard Plug-In Manager.

☞ If you turn on **Update the example design PLLs** option, PLL and phase changes in the MegaWizard Plug-In Manager are reflected automatically in the PLL settings and you do not need to edit the phases with the PLL MegaWizard Plug-In Manager.

Manual Timings Page

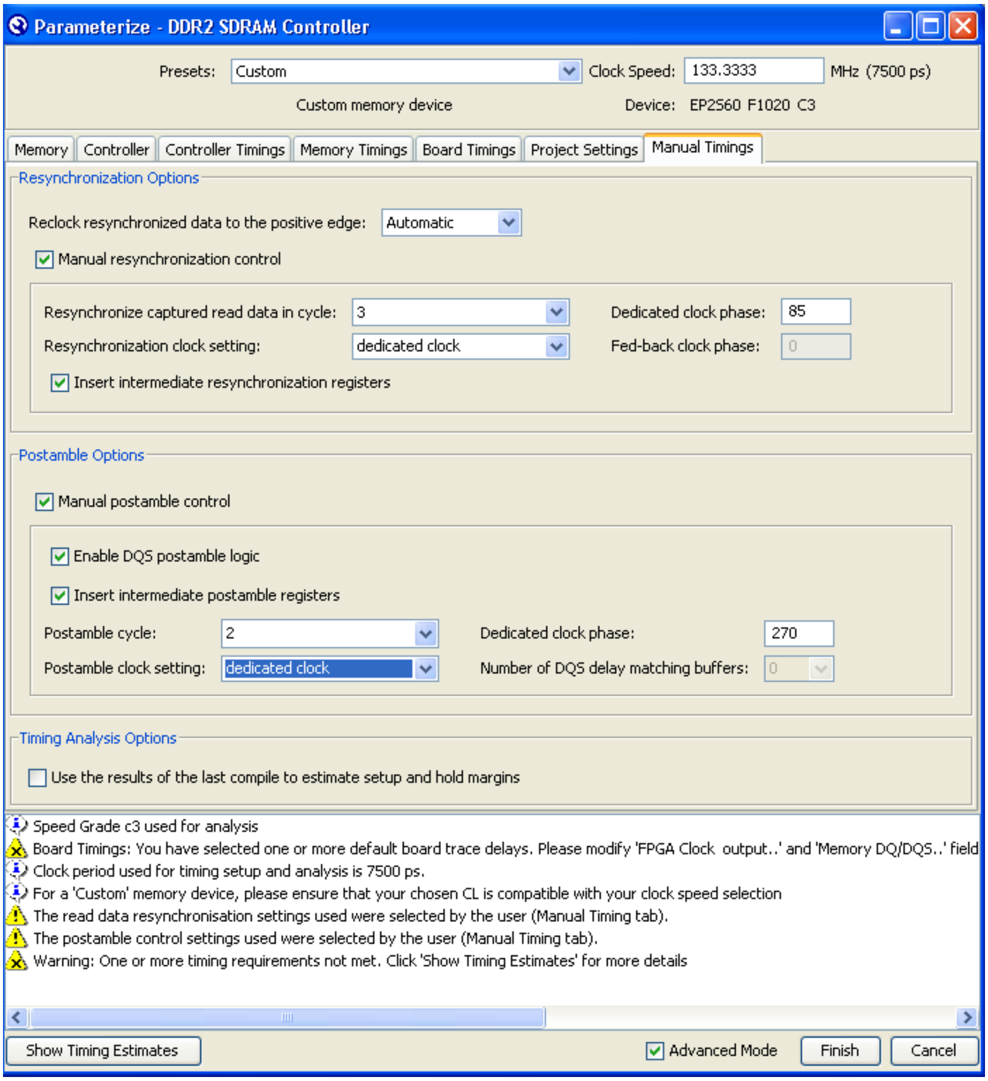
a. Turn on **Manual resynchronization control**.

b. Turn on **Manual postamble control**.

c. Turn on **Enable DQS postamble logic**.

d. Enter the appropriate **Dedicated clock phase** setting in both the Resynchronization Options section, and Postamble Options section as shown in Figure 6.

Figure 6. Manual Timings Page



Presets: Custom
Clock Speed: 133.3333 MHz (7500 ps)
Custom memory device
Device: EP2S60 F1020 C3

Resynchronization Options

Reclock resynchronized data to the positive edge: Automatic

☑ Manual resynchronization control

Resynchronize captured read data in cycle: 3
Dedicated clock phase: 85
Resynchronization clock setting: dedicated clock
Fed-back clock phase: 0

☑ Insert intermediate resynchronization registers

Postamble Options

☑ Manual postamble control

☑ Enable DQS postamble logic

☑ Insert intermediate postamble registers

Postamble cycle: 2
Dedicated clock phase: 270
Postamble clock setting: dedicated clock
Number of DQS delay matching buffers: 0

Timing Analysis Options

☐ Use the results of the last compile to estimate setup and hold margins

Speed Grade c3 used for analysis
Board Timings: You have selected one or more default board trace delays. Please modify 'FPGA Clock output..' and 'Memory DQ/DQS..' field
Clock period used for timing setup and analysis is 7500 ps.
For a 'Custom' memory device, please ensure that your chosen CL is compatible with your clock speed selection
The read data resynchronisation settings used were selected by the user (Manual Timing tab).
The postamble control settings used were selected by the user (Manual Timing tab).
Warning: One or more timing requirements not met. Click 'Show Timing Estimates' for more details

☑ Advanced Mode

Initially, you can input arbitrary numbers for the clock phases and later change the values with the DTW timing analysis script.

Refer to the *DDR/DDR2 SDRAM Controller Compiler User Guide* for more details about any of the page settings.

RTL Code Modifications

After generating the two or three controllers, you must stitch the controllers together to create the top-level file. Figure 1 and Figure 2 show the top-level design of the two- and three- controller designs. The top-level files for both the two- and three- controller designs are provided along with this document for reference, but must be copied and customized by the designer.

Key modifications made to the top-level file (**multiple_ddr2_top.v**) include:

1. The pnf output from all the three controllers are named as separate signals:

    a. `pnf_top_right`

    b. `pnf_top_left`

    c. `pnf_bottom_right`

2. `pnf_per_byte [0:7]` is the OR of the signals `pnf_per_byte_top_right` | `pnf_per_byte_top_left` | `pnf_per_byte_bottom_right`

3. `test_complete` is the OR of the signals `test_complete` = `test_complete_top_right` | `test_complete_top_left` | `test_complete_bottom_right`

4. The PLL and DLL connections are as shown in Figure 1 and Figure 2.

Quartus II Compilation

To achieve maximum performance, you must use the settings shown in Table 4 in the Quartus II software:

Table 4. Key Settings for Maximum Performance in Quartus II (Part 1 of 2)

Page Name	Settings
Analysis and Synthesis Settings	In the Optimization Technique section, select **Speed**.
Fitter Settings	In the Timing-driven compilation section, turn off the **Optimize hold timing** option.
	Turn off the **Optimize fast-corner timing** option.
	In the Fitter Effort section, select **Standard Fit (highest effort).**

Table 4. Key Settings for Maximum Performance in Quartus II (Part 2 of 2)

Page Name	Settings
Timing Analysis Settings	In the Timing analysis processing section, select **Use TimeQuest Timing Analyzer during compilation** or **Use Classic Timing Analyzer during compilation**. *(1)*

Note to Table 4:
(1) Altera recommends that you use the TimeQuest Timing Analyzer.

Refer to Step 4 of the Design Flow section *Add other assignments for the design* in the DDR Timing Wizard User Guide for more details.

I/O Standard Assignments

Source the constraints file **auto_add_constraints.tcl** to assign I/O standards to the top-level pins in the design. This file gets created automatically while generating the multiple controllers, and this file contains references to the **.tcl** files of the controllers used in the design. For the three-controller example, the **auto_add_constraints.tcl** file references the following files:

top_right/add_constraints_for_ddr2_top_right.tcl

bottom_right/add_constraints_for_ddr2_bottom_right.tcl

top_left/add_constraints_for_ddr2_top_left.tcl

Refer to the *DDR and DDR2 SDRAM Controller User Guide* for more information about compiling an example design.

I/O Location Assignments

Assign the top level pins such as address, data, and dqs signals to the closest I/O bank. For example, assign the top level signals of the `top_right` controller to I/O bank 3, the `top_left` controller signals to I/O bank 4, and the `bottom_right` controller signals to I/O bank 7.

Assign `clock_source` to A16, `ddr2_top_right_fedback_clk_in` to A17, and `ddr2_bottom_right_fedback_clk_in` to AM16.

These pins are chosen as they are dedicated clock input pins on the same side of the device as the PLLs they are driving.

Assign all of the `clk_to_sdram` and `sdram_n` pins with the following guidelines:

1. Dedicated clock pin pairs in the same bank as the memory interface resides, for example:

   a. Bank3 = `CLK14p/n` and `CLK15p/n`
   b. Bank4 = `CLK12p/n` and `CLK13p/n`
   c. Bank7 = `CLK6p/n` and `CLK7p/n`
   d. Bank8 = `CLK4p/n` and `CLK5p/n`

2. Dedicated clock pin pairs on the same side of the device as the memory interface resides, for example:
   a. Top Side = `CLK12/13/14/15p` and n
   b. `PLL5_OUT[2..0]` p and n
   c. `PLL11_OUT[2..0]` p and n
   d. Bottom Side = `CLK4/5/6/7p` and n
   e. `PLL6_OUT[2..0]` p and n
   f. `PLL12_OUT[2..0]` p and n

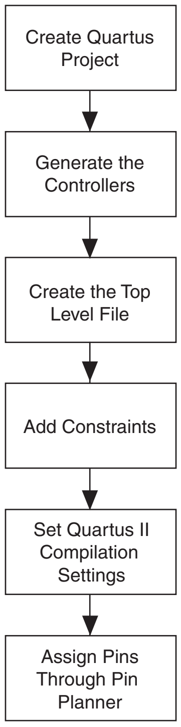
First look for suitable differential pin pairs in the same bank, then in the same side as the controller resides.

Place the pins `reset_n`, `test_complete`, `pnf`, and `pnf_per_byte[7:0]` to the side banks 5 and 6 as these signals should not cause any timing closure issues.

These pins should not be placed too far away from the respective controllers because doing so can cause timing violations.

Figure 7 shows a chart summarizing the steps you have taken thus far to implement a multiple memory interface controller.

Figure 7. Steps to Implement Multiple-Memory Interface Controller


Create Quartus Project
Generate the Controllers
Create the Top Level File
Add Constraints
Set Quartus II Compilation Settings
Assign Pins Through Pin Planner

The next step is to run the DTW (DDR timing wizard).

☞ The timing analysis script (**auto_verify_ddr_timing.tcl**) generated by the Quartus MegaWizard Plug-In Manager does not work with multiple controllers when regional clocks are used. Quartus II software generates the following error messages:

```
Error: Evaluation of Tcl script
auto_verify_ddr_timing.tcl unsuccessful.

Error: Quartus II Shell was unsuccessful. 2 errors, 8
warnings

Error: Quartus II Full Compilation was unsuccessful. 2
errors, 774 warnings
```

For this project, use DTW, as there are many advantages to using it. To learn more about the DTW refer to the *DDR Timing Wizard User Guide*.

Running the DTW Script

To run the DTW script, on the Tools menu, click **Tcl Scripts**. Select **dtw**. Generate a separate ***.dwz** file for each of the controllers with separate names. For example, the three files generated for a three controller design are:

- **top_right_ddr_settings.dwz**
- **top_left_ddr_settings.dwz**
- **bottom_right_ddr_settings.dwz**

Refer to the *DDR Timing Wizard User Guide* for more information about how to use the DTW timing analysis script.

Compile the Design

The DTW script populates the design constraints in the form of an SDC file, and this constraints file needs to be added to the design files before compiling the design.

Refer to the *DDR Timing Wizard User Guide* for more information about how to use the DTW timing analysis script.

DTW Timing Analysis

Now that the design has been compiled with all the constraints, it is necessary to check whether the timing has been met or not.

Refer to the *DDR Timing Wizard User Guide* for details about using the DTW and also how to perform timing analysis.

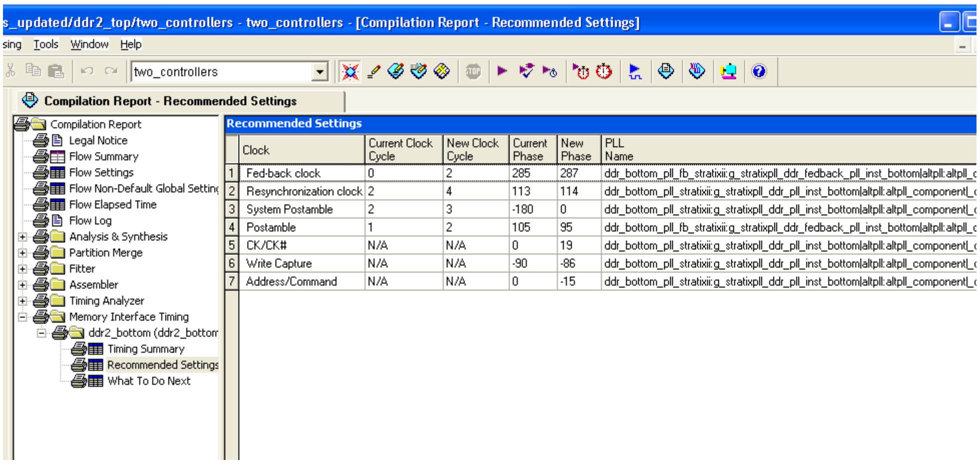
The following steps are performed at the command prompt. The script **dtw_timing_analysis.tcl** file is located in the Quartus II project directory.

```
quartus_sh -t dtw_timing_analysis.tcl -dwz_file
top_right_ddr_settings.dwz

quartus_sh -t dtw_timing_analysis.tcl -dwz_file
top_left_ddr_settings.dwz

quartus_sh -t dtw_timing_analysis.tcl -dwz_file
bottom_right_ddr_settings.dwz
```

After running the script, close and reopen the compilation report to verify the timing analysis results.

The DTW script provides details about what to do next if the timing is not met as shown in Figure 8.

Figure 8. DTW Script Timing Details

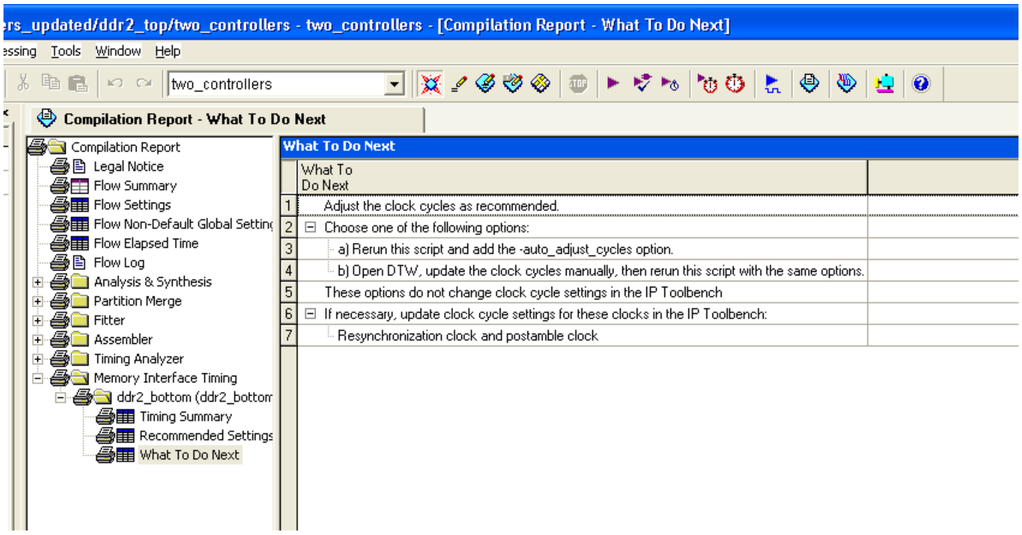


Recommended Settings

	Clock	Current Clock Cycle	New Clock Cycle	Current Phase	New Phase	PLL Name
1	Fed-back clock	0	2	285	287	ddr_bottom_pll_fb_stratixii:g_stratixpll_ddr_fedback_pll_inst_bottom|altpll:altpll_c
2	Resynchronization clock	2	4	113	114	ddr_bottom_pll_stratixii:g_stratixpll_ddr_pll_inst_bottom|altpll:altpll_component|_c
3	System Postamble	2	3	-180	0	ddr_bottom_pll_stratixii:g_stratixpll_ddr_pll_inst_bottom|altpll:altpll_component|_c
4	Postamble	1	2	105	95	ddr_bottom_pll_fb_stratixii:g_stratixpll_ddr_fedback_pll_inst_bottom|altpll:altpll_c
5	CK/CK#	N/A	N/A	0	19	ddr_bottom_pll_stratixii:g_stratixpll_ddr_pll_inst_bottom|altpll:altpll_component|_c
6	Write Capture	N/A	N/A	-90	-86	ddr_bottom_pll_stratixii:g_stratixpll_ddr_pll_inst_bottom|altpll:altpll_component|_c
7	Address/Command	N/A	N/A	0	-15	ddr_bottom_pll_stratixii:g_stratixpll_ddr_pll_inst_bottom|altpll:altpll_component|_c

Typically, cycle adjustment is necessary, as shown in Figure 9.

Figure 9. Cycle Adjustment



What To Do Next

	What To Do Next
1	Adjust the clock cycles as recommended.
2	⊟ Choose one of the following options:
3	a) Rerun this script and add the -auto_adjust_cycles option.
4	b) Open DTW, update the clock cycles manually, then rerun this script with the same options.
5	These options do not change clock cycle settings in the IP Toolbench
6	⊟ If necessary, update clock cycle settings for these clocks in the IP Toolbench:
7	Resynchronization clock and postamble clock

To achieve cycle adjustment, perform the following steps:

```
quartus_sh -t dtw_timing_analysis.tcl -dwz_file
top_right_ddr_settings.dwz -extract_tcos ignore
-read_side tq -auto_adjust_cycles
```

This step is done for the respective controller if the cycles need to be adjusted. After running the DTW script, an indication of what you should do next is provided in the Quartus II software.

After adjusting the cycles, the next step is to adjust the phase of all three clocks. This can be done either in the IP Toolbench or by editing the respective PLL settings.

Once the phase adjustment is done, the next step is to compile the design with the new PLL phases.

```
quartus_sh -t dtw_timing_analysis.tcl -dwz_file
top_right_ddr_settings.dwz -extract_tcos no -read_side
tq -after_iptb import_and_compile
```

Repeat the previous step for all the controllers for which PLL phase adjustment was done. If phase adjustment was done for all three controllers, compilation can be done once and the timing analysis can be done for the three controllers, as shown in the following example:

```
quartus_sh -t dtw_timing_analysis.tcl -dwz_file
top_right_ddr_settings.dwz -extract_tcos no -read_side
tq -after_iptb import_and_compile

quartus_sh -t dtw_timing_analysis.tcl -dwz_file
top_left_ddr_settings.dwz -extract_tcos no

quartus_sh -t dtw_timing_analysis.tcl -dwz_file
bottom_right_ddr_settings.dwz -extract_tcos no
```

In this case, compiling the design three times is avoided.

Trouble-Shooting

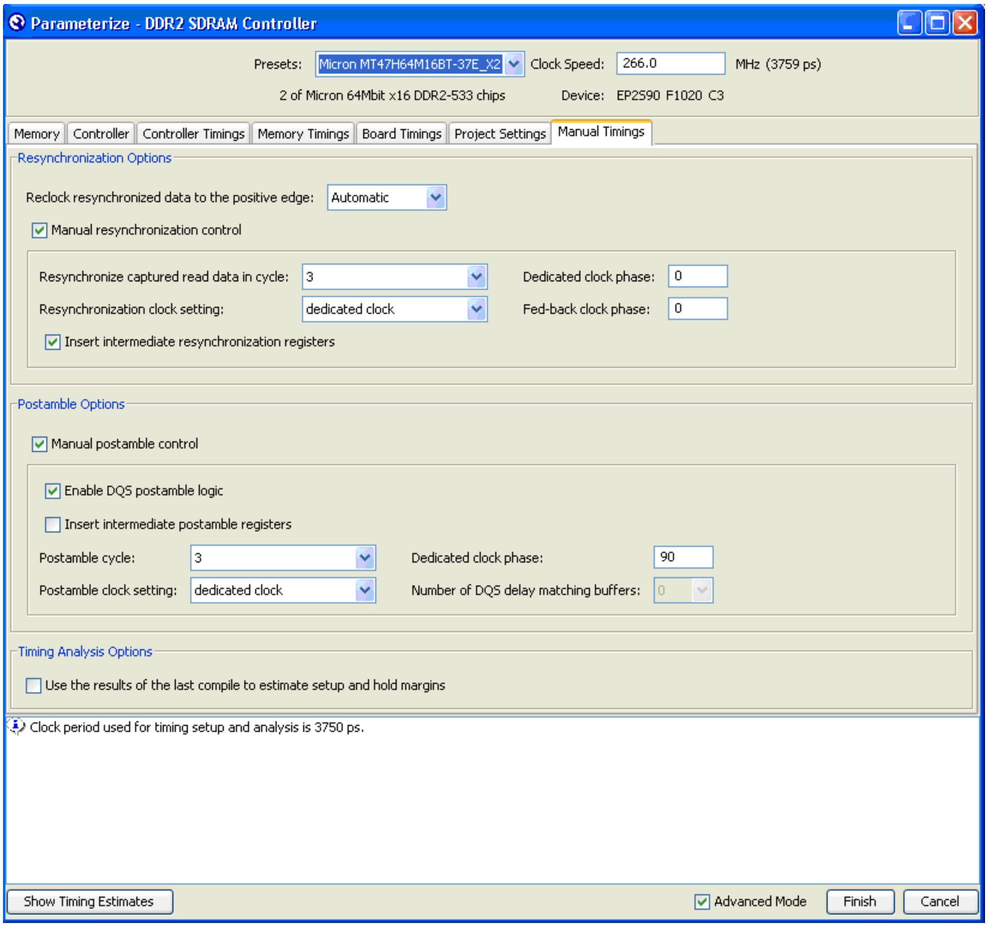
Perform all the steps presented in the DTW script. If there are any violations that relate to destination registers, they appear named `inter_rdata[*]` as shown in "Violations Relating to Destination Registers".

Example 1–1. Violations Relating to Destination Registers

```
ddr2_bottom_right:ddr2_bottom_right_ddr_sdram|ddr2_bottom_right_auk_ddr_s
dram:ddr2_bottom_right_auk_ddr_sdram_inst|ddr2_bottom_right_auk_ddr_datap
ath:ddr_io|ddr2_bottom_right_auk_ddr_dqs_group:\g_datapath:1:g_ddr_io|int
er_rdata[5]
```

If you receive an `inter_rdata[*]` violation, under Manual Page Settings, toggle the **Insert intermediate synchronization registers** option. If the setting is on, turn it off, if it is off, turn it on. Figure 10 shows this option turned on.

Figure 10. Insert Intermediate Resynchronization Registers Unchecked


Parameterize - DDR2 SDRAM Controller
Memory | Controller | Controller Timings | Memory Timings | Board Timings | Project Settings | Manual Timings
Show Timing Estimates
Advanced Mode
Finish
Cancel

Presets: Micron MT47H64M16BT-37E_X2 Clock Speed: 266.0 MHz (3759 ps)

2 of Micron 64Mbit x16 DDR2-533 chips Device: EP2S90 F1020 C3

Resynchronization Options

Reclock resynchronized data to the positive edge: Automatic

☑ Manual resynchronization control

Resynchronize captured read data in cycle: 3 Dedicated clock phase: 0

Resynchronization clock setting: dedicated clock Fed-back clock phase: 0

☑ Insert intermediate resynchronization registers

Postamble Options

☑ Manual postamble control

☑ Enable DQS postamble logic

☐ Insert intermediate postamble registers

Postamble cycle: 3 Dedicated clock phase: 90

Postamble clock setting: dedicated clock Number of DQS delay matching buffers: 0

Timing Analysis Options

☐ Use the results of the last compile to estimate setup and hold margins

Clock period used for timing setup and analysis is 3750 ps.

☑ Advanced Mode

Conclusion

As shown in this application note, there are multiple ways to create multiple-controller designs using the DDR/DDR2 SDRAM controller. This application note described the steps necessary to create and meet the performance requirements for two types of multiple-controller designs. If your designs are different than those described in this application note, you must follow the guidelines and make appropriate changes wherever necessary. Knowing the requirements and limitations for the multiple-memory interface up front allows you to architect your system better.

Document Revision History

Table 5 shows the revision history for this document.

Table 5. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
July 2007 v2.0	Rewrote introduction, removed the following sections (and associated subsections) -- Set Up the Project, Launch the DDR SDRAM Controller MegaCore Function, Edit the Top- Level Design File, Add the Constraints, Appendix. Replaced removed sections with the following sections (sub sections not listed here, see document) -- Design Details, Design Considerations, DLL Resource Considerations, PLL Resource Considerations, Instantiating DDR/DDR2 SDRAM Memory Controllers, RTL Code Modifications, Quartus II Compilation, I/O Standard Assignments, I/O Location Assignments, Running the DTW Script, Compile the Design, DTW Timing Analysis, Trouble-Shooting, Conclusion. Replaced all pictures with a new series of pictures.	—
August 2005 v1.0	Initial Release	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/
Literature Services:
literature@altera.com

I.S. EN ISO 9001