LogiCORE IP DDR2 Memory Controller for PowerPC 440 Processors (v3.00c)

DS567 March 1, 2011 **Product Specification**

Introduction

This data sheet describes the LogiCORE™ IP DDR2 Memory Controller reference design for the PowerPC® 440 processor block embedded in the Virtex-5 FXT Platform FPGAs. The processor block interfaces with the Memory Controller Interface (MCI) and provides the control interface for DDR2 memory.

Features

- Supports a maximum performance of 333 MHz in the fastest speed grade
- Supports 16-bit, 32-bit, and 64-bit data widths, and 72-bit data width with ECC (DQ:DQS = 8:1)
- Supports DDR2 SDRAM single-rank registered DIMMs and components
- Supports the following DDR2 SDRAM features:
  - CAS latencies (3, 4, 5)
  - Additive latencies (0, 1, 2, 3, 4)
  - On-die termination (ODT)
  - Burst lengths (4, 8)
- Supports bank management (up to four banks open)
- Performs the memory device initialization sequence upon power-up
- Performs auto-refresh cycles

LogiCORE IP Facts	
Core Specifics	
Supported Device Family (1)	Virtex-5
Resources Used	See Table 9.
Special Features	None
Provided with Core	
Documentation	Product Specification
Design File Formats	Verilog
Constraints File	UCF - in EDK Pcore Directory
Verification	Verilog Test Bench
Instantiation Template	Verilog Wrapper
Additional Items	None
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 12.1
Verification	Mentor Graphics® ModelSim® v6.5c and above
Simulation	Mentor Graphics ModelSim v6.5c and above
Synthesis	Synopsys® Synplify Pro® D-2009.12
Support	
Provided by Xilinx, Inc.	

1. For a complete list of supported devices, see the 12.1 release notes for this core.

Functional Description

The PPC440MC DDR2 Memory Controller interfaces directly to the PowerPC processor through the MCI (see Figure 1). To achieve hardware functionality and maximum performance with the memory controller interface, use the relevant optimal UCF provided in the EDK PCORE directory. There is only one optimal UCF for each device/package/processor combination.

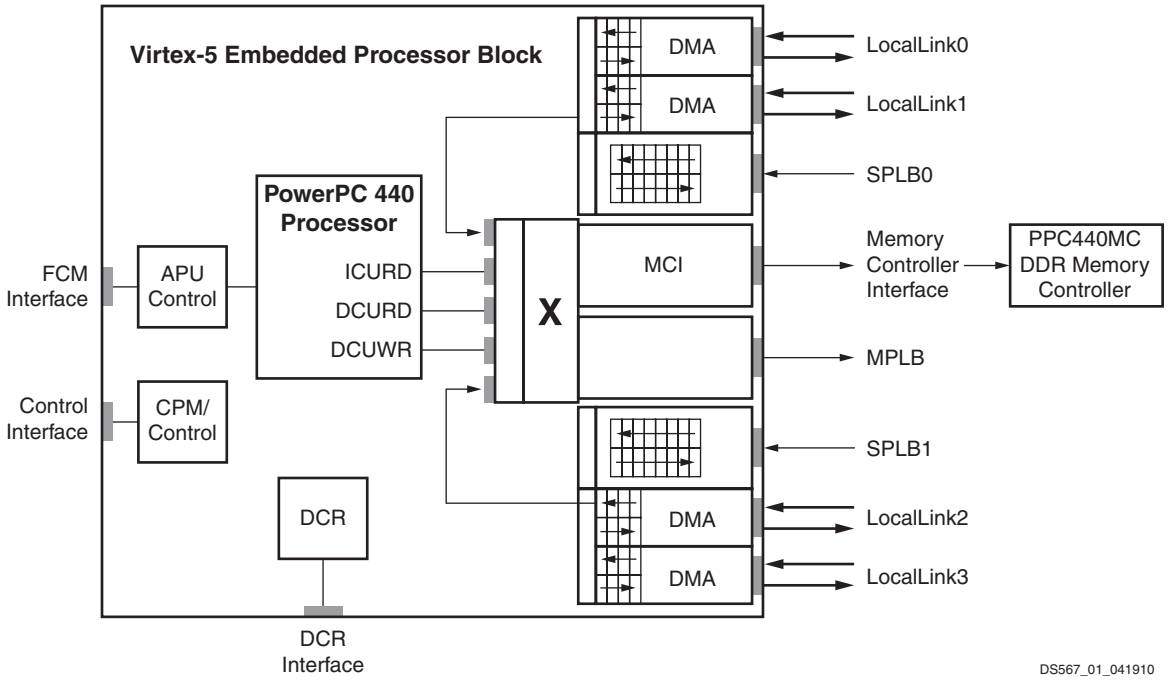

Figure 1: **PPC440 MCI and PPC440MC DDR2 Memory Controller Block Diagram**

I/O Signals

The PPC440MC DDR2 Memory Controller signals are listed and described inTable 1.

Table 1: **PPC440MC DDR2 Memory Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Initial Status	Description
PPC440 MCI Signals				
MIMCREADNOTWRITE	MCI	I		This signal indicates if the operation is a read or a write.
MIMCADDRESS[0:35]	MCI	I		Address bus
MIMCADDRESSVALID	MCI	I		When asserted, this signal indicates the data on the address bus is valid.
MIMCWRITEDATA[0:127]	MCI	I		Data bus
MIMCBYTEENABLE[0:15]	MCI	I		Byte enable for the data on the data bus
MIMCWRITEDATAVALID	MCI	I		When asserted, this signal indicates the data on the data bus is valid.
MIMCBANKCONFLICT	MCI	I		This signal is asserted if the bank being accessed is different from the bank accessed in the previous command.

Table 1: **PPC440MC DDR2 Memory Controller I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Initial Status	Description
MIMCROWCONFLICT	MCI	I		This signal is asserted if the row being accessed is different from the row accessed in the previous command.
MCMIREADDATA[0:127]	MCI	O		Read data bus
MCMIREADDATAVALID	MCI	O		When asserted, this signal indicates the data on the read data bus is valid.
MCMIREADDATAERR	MCI	O		This signal is asserted when an uncorrectable error is detected by the ECC logic.
MCMIADDRREADYTOACCEPT	MCI	O		This signal is asserted when the PPC440MC DDR2 Memory Controller is ready to accept transactions.
DDR2 Signals				
DDR2_DQ (C_DDR_DWIDTH – 1:0)	DDR2	I/O		DDR2 data bus
DDR2_DQS (C_DDR_DQS_WIDTH – 1:0)	DDR2	I/O		DDR2 data strobe
DDR2_DQS_N (C_DDR_DQS_WIDTH – 1:0)	DDR2	I/O		DDR2 inverted data strobe
DDR2_A (C_DDR_RAWIDTH – 1:0)	DDR2	O		DDR2 address
DDR2_BA (C_DDR_BAWIDTH – 1:0)	DDR2	O		DDR2 bank address
DDR2_RAS_N	DDR2	O		DDR2 row address strobe
DDR2_CAS_N	DDR2	O		DDR2 column address strobe
DDR2_WE_N	DDR2	O		DDR2 write enable
DDR2_CS_N (C_NUM_RANKS_MEM – 1 down to 0)	DDR2	O		DDR2 chip selects
DDR2_ODT (C_DDR2_ODT_WIDTH – 1:0)	DDR2	O		DDR2 ODT enable signal
DDR2_CKE (C_DDR2_NUM_RANKS_MEM – 1:0)	DDR2	O		DDR2 clock enable signal
DDR2_DM (C_DDR_DM_WIDTH – 1:0)	DDR2	O		DDR2 data mask
DDR2_CK (C_NUM_CLK_PAIRS – 1:0)	DDR2	O		DDR2 clock
DDR2_CK_N (C_NUM_CLK_PAIRS – 1:0)	DDR2	O		DDR2 inverted clock
System Clock and Reset Signals				
MC_MIBCLK	CLK	I		Clock
MI_MCCLK90	CLK	I		MC_MIBCLK phase shifted by 90
MI_MCCLKDIV2	CLK	I		MC_MIBCLK divided by 2
MI_MCCLK_200	CLK	I		IDELAY reference clock
MI_MCRESET	RST	I		Reset

Table 1: **PPC440MC DDR2 Memory Controller I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Initial Status	Description
IDELAY_CTRL_RDY_I	FPGA	I		Used to daisy chain to downstream IDELAYCTRL when sharing IDELAYCTRLs between multiple pcores. This input port must be tied to "1" when not sharing IDELAYCTRLs.
IDELAY_CTRL_RDY	FPGA	O		This output from an upstream IDELAYCTRL pcore connects to the downstream IDELAY_CTRL_I input port when sharing IDELAYCTRLs between multiple pcores.

PPC440MC DDR2 Memory Controller Design Parameters

To create a uniquely tailored PPC440MC DDR2 Memory Controller, certain parameterizable features in the PPC440MC DDR2 Memory Controller design allow a design that only utilizes the resources required by the system and runs at the best possible performance. Table 2 lists the parameterizable features in the PPC440MC DDR2 Memory Controller

.

Table 2: **PPC440MC DDR2 Memory Controller Design Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Value
PPC440MC DDR2 Memory Controller Features			
Base Address for Memory	C_MEM_BASEADDR		32'hFFFFFFFF
High Address for Memory	C_MEM_HIGHADDR		32'h00000000
CPMINTERCONNECTCLK to PPC440MC DDR2 clock ratio	C_MIB_MC_CLOCK_RATIO	0 or 1 for 1:1, 2:1, or 3:1clock ratios 2 for 3:2 clock ratio	0
DDR2 clock period (t_{CK}) in ps	C_MC_MIBCLK_PERIOD_PS	3000 to 8000	3000
Number of generated clock pairs supplied to the DDR2 memory	C_NUM_CLK_PAIRS	1 to 5	1
Supported number of external DDR2 memory ranks	C_NUM_RANKS_MEM	1	1
Include ECC Logic	C_INCLUDE_ECC_SUPPORT	0 = Disable 1 = Enable	0
DDR2 Features			
DDR2 Data Width	C_DDR_DWIDTH (1)	16, 32, 64, 72	64
DDR2 Strobe Width	C_DDR_DQS_WIDTH	2, 4, 8, 9	8
DDR2 Data Mask Width	C_DDR_DM_WIDTH	2, 4, 8, 9	8
DDR2 Row Address Width	C_DDR_RAWIDTH	All supported memory row address widths	14
DDR2 Column Address Width	C_DDR_CAWIDTH	All supported memory column address widths	10
DDR2 Bank Address Width	C_DDR_BAWIDTH	2, 3	2
DDR2 CAS Latency	C_DDR_CAS_LAT (2)	3, 4, 5	5
DDR2 Burst Length	C_DDR_BURST_LENGTH (3)	4, 8	4
DDR2 is a registered DIMM	C_REG_DIMM	0 = Unbuffered memory 1 = Registered memory	0

Table 2: **PPC440MC DDR2 Memory Controller Design Parameters *(Cont'd)***

Feature/Description	Parameter Name	Allowable Values	Default Value
On Die Termination Selection	C_DDR2_ODT_SETTING	0 = Disables ODT 1 = ODT enabled, R_{TT} = 75Ω 2 = ODT enabled, R_{TT} = 150Ω 3 = ODT enabled, R_{TT} = 50Ω	1
DDR2 ODT Width	C_DDR2_ODT_WIDTH	0 to 4	1
DDR2 Additive Latency	C_DDR2_ADDT_LAT	0 to 4	0
Delay after ACTIVE command before READ/WRITE command (ps)	C_DDR_TRCD		15000
Delay after ACTIVE command before PRECHARGE command (ps)	C_DDR_TRAS		40000
Delay after PRECHARGE command (ps)	C_DDR_TRP		15000
Delay after AUTOREFRESH before another command (ps)	C_DDR_TRFC		70000
Read to PRECHARGE command delay (ps)	C_DDR_TRTP		7500
Write Recovery Time (ps)	C_DDR_TWR		15000
Write-to-Read Command Delay (ps)	C_DDR_TWTR		10000
Average periodic refresh command interval (ns)	C_DDR_TREFI		7800
Skip 200 µs power up delay for simulation	C_SIM_ONLY	0, 1	0
IDELAY high-performance mode	C_IDEL_HIGH_PERF	TRUE, FALSE	TRUE
$\log_2$ of C_DQS_WIDTH	C_DQS_BITS	—	3
$\log_2$ of C_DDR_DWIDTH	C_DQ_BITS	—	6
$\log_2$ of C_NUM_RANKS_MEM	C_CS_BITS	—	0
Optional pipeline stage in read data path	C_READ_DATA_PIPELINE	0 or 1	0
Number of IDELAYCTRLs internally generated	C_NUM_IDELAYCTRL	-	1
Reduces drive strength of DDR2 SDRAM I/O when set to 1	C_REDUCE_DRV	0,1	0
Applies IODELAY_GROUP constraint to all IDELAYCTRL and all IDELAY instances in pcore	C_IODELAY_GRP	STRING	<InstanceName>
FPGA device speed grade required for predictable IP	C_FPGA_SPEED_GRADE	1,2, 3	2

1. – MCI burst width of 32 for DDR2 data width of 16
   – MCI burst width of 64 for DDR2 data width of 32
   – MCI burst width of 128 for DDR2 data width of 64/72
2. When the parameter C_DDR_CAS_LAT is set to 3, the user must set the C_DDR2_ADDT_LAT parameter to any value from 1 to 4.
3. – MCI burst length of 2 for DDR2 burst length of 4
   – MCI burst length of 4 for DDR2 burst length of 8

Allowable Parameter Combinations

The PPC440MC DDR2 Memory Controller allows one external rank of memory. Individual rank address ranges are calculated by C_MEM_BASEADDR and C_MEM_HIGHADDR. The ranges must comprise a complete, contiguous power of two range such that range = 2^m, and the m least-significant bits of C_MEM_BASEADDR must be zero.

All external memory ranges are calculated contiguous to each other in the system addressable space. In addition, all external memory ranks are identical in size to other ranks of memory space.

Parameter - Port Dependencies

Table 3 illustrates the port and parameter dependencies. With the configuration flexibility of the channelized access memory controller, the user must be aware of unused I/Os based on the configuration of each port. The designer must also be aware of the external memory interface signals based on the type of memory utilized in the system.

Table 3: **Parameter-Port Dependencies**

Name	Affects	Depends	Relationship Description
Design Parameters			
C_NUM_CLK_PAIRS	DDR_Clk, DDR_Clkn		Affects size of signals.
C_NUM_RANKS_MEM	DDR_CKE, DDR_CSn, DDR_ODT, C_MEM_BASEADDR, C_MEM_HIGHADDR		Affects size of signals.
C_DDR_DWIDTH	DDR_DQ		Parameter affects size of DDR DQ signals.
C_DDR_DQS_WIDTH	DDR_DQS, DDR_DQS_N		Parameter affects size of DDR DQS signals.
C_DDR_DM_WIDTH	DDR_DM		Parameter affects size of DDR DM signals.
C_DDR_RAWIDTH	DDR_Addr		Affects size of signal.
C_DDR_BAWIDTH	DDR_BankAddr		Affects size of signal.
I/O Signals			
DDR_Clk, DDR_Clkn		C_NUM_CLK_PAIRS	Size depends on C_NUM_CLK_PAIRS parameter.
DDR_CKE, DDR_CSn		C_NUM_RANKS_MEM	Size depends on C_NUM_RANKS_MEM parameter.
DDR_DQ		C_DDR_DWIDTH	Size depends on parameter setting.
DDR_DQS		C_DDR_DQS_WIDTH	Size depends on parameter setting.
DDR_DM		C_DDR_DM_WIDTH	Size depends on parameter setting.
DDR_Addr		C_DDR_RAWIDTH	Size depends on parameter setting.
DDR_BankAddr		C_DDR_BAWIDTH	Size depends on parameter setting.
DDR_DQS_N, DDR_ODT		C_NUM_MEM_RANKS	Size depends on parameter setting.

Connecting to Memory

DDR2 SDRAM accesses can be halfword (two bytes), word (four bytes), or doubleword (eight bytes), depending on the user configuration. Data to and from the MCI to the PPC440MC DDR2 Memory Controller is organized as big-Endian (D[0:63], D0 is the MSB). The PPC440MC DDR2 Memory Controller is organized as little-Endian (D[63:0], D63 is the MSB). Table 4 shows the interconnection from the MCI to the PPC440MC DDR2 Memory Controller and from the PPC440MC DDR2 Memory Controller to the DDR2 memory.

Table 4: **MCI to PPC440MC DDR2 to DDR2 Signal Connections**

Description	MCI to PPC440MC DDR2 Signal [MSB:LSB]	PPC440MC DDR2 to DDR2 Signal [MSB:LSB]
Data Bus	MIMCWRITEDATA[0:(C_DDR_DWIDTH * 2) – 1] MCMIREADDATA[0:(C_DDR_DWIDTH * 2) – 1]	DDR2_DQ[C_DDR_DWIDTH – 1:0]
Address Bus	MIMCADDRESS[0:35]	DDR2_A[C_DDR_AWIDTH – 1:0]
Bank Address	Memory Controller Generated	DDR2_BA[C_DDR_BAWIDTH – 1:0]
Data Strobe	Memory Controller Generated	DDR2_DQS[(C_DDR_DWIDTH /C_DDR_DQS_WIDTH) – 1:0]
Data Mask	MIMCBYTEENABLE[0:(C_MIBDATA_WIDTH/8) – 1]	DDR2_DM[(C_DDR_DWIDTH /C_DDR_DM_WIDTH) – 1:0]

Memory Part Parameter Considerations

Many parameters of the PPC440MC DDR2 Memory Controller must be set according to the memory device used for proper operation. These parameters include control port widths and memory timing specifications. The values for these parameters can usually be found in memory vendor data sheets or other documentation though may be labeled slightly differently.

MIG PHY UCF and MHS Considerations

This core utilizes the MIG controller PHY to calibrate memory. The MIG PHY requires a set of pinout-specific placement and timing constraints to operate. Special handling of IDELAYCTRLs may also be necessary when multiple IDELAY-enabled IP cores share an I/O bank and clock region.

Using Optimal Pinouts

This sections details designs that use the recommended high performance pinouts provided in <EDK_Install_Dir>/hw/XilinxProcessorIPLib/pcores/ppc440mc_ddr2_<version>/data/ppc440mc_ddr2_ucfs.

1. Choose the applicable predefined UCF from the directory above. The UCF file names are organized in the following order: maximum total memory DQ width, part name, part package, and I/O banks occupied. Multiple UCFs are provided in some part/package combinations to support both PPC440 processors.
2. Copy the predefined UCF into the project UCF file, often named data/system.ucf.
3. If necessary, modify the instance names of the copied UCF to match top-level port names.
4. If necessary, modify the hierarchy of the copied UCF constraints to match the actual design hierarchy. Specifically, "u_ddr2_top" will need to be modified to the instance name of the PPC440MC_DDR2 core name, perhaps with a "*" wildcard added to the start of the name to match additional user design hierarchy.

Using Custom MIG-Compatible Pinouts

It is strongly recommended to use the provided I/O pinouts for optimal performance. However, MIG-compatible custom pinouts can be generated and used with the PPC440MC_DDR2 core. To use a custom pinout, the MIG "Update Design" flow must be followed to generate the required UCF and MHS constraints. The steps are:

1. A MIG 3.2 project must be generated through Xilinx CORE Generator GUI. MIG is located in the "Memories & Storage Elements->Memory Interface Generators" section. To access all available MIG versions, select the "All IP Versions" check box in the Xilinx CORE Generator GUI. Note: Because the desired pinout is already available, all banks can be selected in the MIG bank selection screen.
2. If the default MIG pinout generated in <MIG Project>/user_design/par/ddr2_sdram.ucf is not the desired pinout, run the MIG > Update Design flow using the MIG project and a UCF with the desired pinout in a MIG-formatted UCF. Note: The input UCF parsing to the MIG > Update Design flow currently requires strict adherence to the formatting similar to MIG UCF output for complete Update Design UCF parsing. Warnings in this step relating to missing pins in the UCF can be ignored if similar pins are not used by the PPC440MC_DDR2 core. For example the clk200_p, clk200_n, sys_rst_n, phy_init_done, and extra ddr2_cs_n, ddr2_odt, ddr2_ck, and ddr_ck_n ports can be ignored during this step.
3. Copy the generated MIG UCF from <MIG Project>/user_design/par/ddr2_sdram.ucf into the project UCF file, often named data/system.ucf. The following steps involving UCFs will modify this file.
4. If necessary, modify the instance names of the copied UCF to match top-level port names.
5. In the UCF, rename "u_usr_rd" to "usr_rd" in the following line: INST "*/u_usr_rd/gen_rden_sel_mux*.u_ff_rden_sel_mux" TNM = "TNM_RDEN_SEL_MUX";
6. The maximum memory speed of a custom pinout is generally determined by static timing analysis, up to 333MHz in the fastest speed grade. To improve timing results, the C_READ_DATA_PIPELINE MHS parameter may need to be set to '1'.

More information on the MIG Update Design flow can be found in the MIG documentation, and in Xilinx Answer 29313.

Multiple PPC440MC_DDR2 Controllers

When using 2 PPC440MC_DDR2 controllers in a single FPGA, a pinout-specific UCF is required for each controller. In addition, all of the instance names, time groups, and TIMESPECS in the project UCF must be made unique to each controller. For the instance (INST) portion of each constraint, an additional level of hierarchy can be added to the start of the instance name. The time group (TNM) and TIMESPEC constraints must also be given unique labels. For example, the following constraints for a controller instance name 'dimm0' could be changed from:

```
INST "*/u_phy_calib/gen_rd_data_sel*.u_ff_rd_data_sel" TNM = "TNM_RD_DATA_SEL";
TIMESPEC "TS_MC_RD_DATA_SEL" = FROM "TNM_RD_DATA_SEL" TO FFS
"TS_SYS_CLK" * 4;
```

To:

```
INST "*/dimm0*/u_phy_calib/gen_rd_data_sel*.u_ff_rd_data_sel" TNM = "TNM_RD_DATA_SEL_DIMM0";
TIMESPEC "TS_MC_RD_DATA_SEL_DIMM0" = FROM "TNM_RD_DATA_SEL_DIMM0" TO FFS
"TS_SYS_CLK" * 4;
```

Similar changes must be made to all other constraints for each controller UCF.

IDELAYCTRL Shared Pinout Considerations

Note: When the PPC440MC_DDR2 does not have I/O locations in the same clock region as another IP core which uses IODELAYs, this section can be ignored. In Virtex-5, I/O banks fully contain multiple clock regions, so that I/O in different banks also exist in different clock regions. This makes a simple test of a pinout to check for the non-sharing of clock regions.

To utilize the dynamic delay elements available in Virtex-5, the PPC440MC_DDR2 uses IODELAY and IDELAYCTRL elements to improve read timing margin. Multiple other Xilinx IP cores and user custom logic also utilize IODELAY and IDELAYCTRL elements which may need to be coordinated with the PPC440MC_DDR2 core. Single PPC440MC_DDR2 designs or multiple cores whose I/O do not share clock regions are automatically handled by the Xilinx implementation tools and do not require this section.

A single IDELAYCTRL element calibrates all IODELAYs in a clock region. When another IODELAY-enabled IP core shares the clock region with the PPC440MC_DDR2 core I/O for its pin placement, a single IDELAYCTRL must be shared between both cores. This can be accomplished by allowing a single "master" core to instantiate the IDELAYCTRL which the other "slaves" utilize. The slave cores must not instantiate any IDELAYCTRLs. The master core outputs the RDY signal from its instantiated IDELAYCTRL which the slaves can optionally use to trigger to start their use of their IODELAYs after FPGA programming or IDELAYCTRL reset. Sharing of IDELAYCTRLs are signified to the implementation tools by all shared IP cores having the same IODELAY_GROUP constraint value on all of their IDELAYCTRL and IODELAY elements. The PPC440MC_DDR2 provides the C_IODELAY_GRP and C_NUM_IDELAYCTRL parameters and IDELAY_CTRL_RDY_I and IDELAY_CTRL_RDY ports to provide this functionality.

I/O Clock Region Sharing Rules

1. - All IDELAYCTRL and IDELAY elements must have the same IODELAY_GROUP constraint value. This is provided by setting the PPC440MC_DDR2 C_IODELAY_GRP MHS parameter. Other cores which do not provide explicit top-level IODELAY_GROUP parameters may be set with IODELAY_GROUP constraints on all IDELAYCTRL and IODELAYs instantiated by the core in the UCF file. By default EDK sets the C_IODELAY_GRP parameter of each PPC440MC_DDR2 to the unique instance name in the MHS, generally allowing single or non-shared clock region pinouts between multiple cores to function without any special handling.
2. - Only one IP core should have IDELAYCTRLs instantiated within it. The PPC440MC_DDR2 provides this with the C_NUM_IDELAYCTRL MHS parameter. Setting to '0' will prevent the instantiation of an IDELAYCTRL.
3. - Resetting the IP core which contains the IDELAYCTRLs will affect the IODELAYs of all IP cores in the IODELAY_GROUP. All cores in the group should be reset with the same reset signal.
4. - The PPC440MC_DDR2 provides the IDELAY_CTRL_RDY port as the output AND of all of its IDELAYCTRLs RDY signals and its IDELAY_CTRL_RDY_I input. The IDELAY_CTRL_RDY signal can be consumed by other IP cores that are using the IDELAYCTRL to indicate when IDELAYCTRL calibration is completed and the core can use its IODELAYs.
5. - The PPC440MC_DDR2 provides the IDELAY_CTRL_RDY_I signal to receive the RDY signal from another core's IDELAYCTRL, such as another PPC440MC_DDR2 IDELAY_CTRL_RDY port. The IDELAY_CTRL_RDY_I is AND'd with the internal core reset before allowing memory calibration to begin. The port is defaulted to '1' by the EDK tools when unconnected. Setting to '0' will prevent the PPC440MC_DDR2 from beginning operation.

Shared Clock Region Example

This example MHS excerpt contains the relevant parameters for two PPC440MC_DDR2 cores with I/O ports LOCed to the same bank/clock region. The first memory is the master and provides the RDY signal to the second core, which does not have any IDELAYCTRLs generated.

```
BEGIN ppc440mc_ddr2
 PARAMETER INSTANCE = DDR2_SDRAM_DIMM0
 PARAMETER C_IODELAY_GRP = MYGRP
 PARAMETER C_NUM_IDELAYCTRL = 1
 PORT idelay_ctrl_rdy = RDY_MYGRP
END
BEGIN ppc440mc_ddr2
 PARAMETER INSTANCE = DDR2_SDRAM_DIMM1
 PARAMETER C_IODELAY_GRP = MYGRP
 PARAMETER C_NUM_IDELAYCTRL = 0
 PORT idelay_ctrl_rdy_i = RDY_MYGRP
END
```

UCF Constraint Prorating for Slower Clock Frequencies

The optimal constraint files provided with the PPC440MC_DDR2 assume a 333MHz clock speed. When running at a slower memory rate, the UCF constraints can be relaxed to ease timing closure.

PERIOD Constraint

The PERIOD constraint in the UCF should be removed if the memory clock, mc_mibclk, is already covered by an existing system constraint. This typically occurs when a PERIOD constraint is placed on the system clock input and automatically propagates through DCMs and PLLs to constrain the memory clock.

Multi-cycle Constraints

The multi-cycle constraints which end in "TS_SYS_CLK" * 4 are dependent on the memory clock frequency. The total value should not exceed 4 memory clock (mc_mibclk) cycles. For simplicity, the "TS_SYS_CLK" * 4 TIMESPEC value can be replaced with an integer time value corresponding to four memory clock cycles. As an example, a 200MHz memory clock design can replace the "TS_SYS_CLK" * 4 clauses with 20ns.

DQS Gate MAXDELAY Constraints

The MAXDELAY constraint involving the "en_dqs_sync" net can have its requirements relaxed based on the actual memory clock frequency. The following constraint can be prorated if the memory clock is slower than 333MHz:

```
NET "*/u_phy_io/gen_dqs*.u_iob_dqs/en_dqs_sync" MAXDELAY = 800 ps;
```

Some examples for "en_dqs_sync" timespec values by clock frequency: 300MHz = 850ps, 267MHz = 900ps and 200MHz = 950ps.

Note that the similar "en_dqs" MAXDELAY constraint value should not be modified with frequency.

PPC440 CPU/MCI Considerations

This section details the connection of the PPC440MC DDR2 Memory Controller to the PPC440 Memory Controller Interface (MCI) and the PPC440 MCI parameter settings.

PPC440 MCI Bus Connections

Connecting signals between the PPC440MC DDR2 Memory Controller and the PPC440 core instances is accomplished through a single bus connection, PPC440MC.

PPC440 Clocking

The CPMMCCLK clock port of the PPC440 instance should usually be the same clock as the MC_MIBCLK port of the PPC440MC DDR2 Memory Controller.

PPC440 C_PPC440MC_CONTROL Settings

The MI_CONTROL register configures the PPC440 MCI with the correct data widths, latency, burst lengths, and row/bank management behavior. MI_CONTROL is the PPC440 DCR register, while an initial value can be provided through the C_PPC440MC_CONTROL parameter on the PPC440 MHS instance. Table 5 shows the suggested bit settings for the C_PPC440MC_CONTROL parameter.

Table 5: **C_PPC440PC_CONTROL/MI_CONTROL Register Settings**

Bit	PPC440 Label	Allowable Values
0	enable	1
1	Rowconflictholdenable	1
2	Bankconflictholdenable	1
3	Directionconflictholdenable	1
4:5	Autoholdduration	10
6	2:3 clock Ratio mode	0 = Integer CPMMCCLK-CPMINTERCONNECTLCK clock ratio 1 = 2:3 CPMMCCLK:CPMINTERCONNCECTCLK clock ratio
7	overlapdwr	0
8:9	Burstwidth	Double the DDR memory DQ data width because MCI must provide 2x the DQ width every clock period 00 = 128 bit 01 = 64 bit 11 = 32 bit
10:11	Burstlength	Half of C_DDR_BURST_LENGTH as 2x DQ width is transferred each MCI clock cycle 01 = C_DDR_BURST_LENGTH set to 4 10 = C_DDR_BURST_LENGTH set to 8
12:15	Write Data Delay (WDD)	0000
16	Read Modify Write (RMW)	0 = Disable ECC. C_INCLUDE_ECC_SUPPORT set to 0 1 = Enable ECC. C_INCLUDE_ECC_SUPPORT set to 1
17:23	Reserved	0000000
24	PLB Priority Enable	0 = SPLB0/SPLB1 master priority disabled 1 = SPLB0/SPLB1 master priority enabled (default)
25:27	Reserved	000
28	Pipelined Read Enable	1

Table 5: **C_PPC440PC_CONTROL/MI_CONTROL Register Settings *(Cont'd)***

Bit	PPC440 Label	Allowable Values
29	Pipelined Write Enable	1
30:31	Reserved	11

Address Mapping

Table 6 shows the address mapping from the MCI interface. The address on the MCI interface maps to a continuous address space at the memory. The memory address mapping is determined by the data width of the memory, since each column address maps a location in memory for the width of each data beat.

Table 6: **Physical Controller Address Mapping**

Memory Use	Physical Controller Address
Address Offset	$\log_2$(C_DDR_DWIDTH/8)
Column Address	MIMCADDRESS [(C_DDR_CAWIDTH + Address Offset – 1) : Address Offset]
Row Address	MIMCADDRESS [(C_DDR_RAWIDTH + C_DDR_CAWIDTH + Address Offset – 1) : (C_DDR_CAWIDTH + Address Offset)]
Bank Address	MIMCADDRESS [(C_DDR_BAWIDTH + C_DDR_RAWIDTH + C_DDR_CAWIDTH + Address Offset – 1) : (C_DDR_RAWIDTH + C_DDR_CAWIDTH + Address Offset)]

An address offset is calculated based on the width of the DDR2 data bus. The DDR2 column address locations are calculated based on the offset, followed by the row address and bank address. Table 7 shows the bit locations for the DDR2 address bus. The address, which is from the MCI, is in big-Endian format.

Table 7: **Calculation of Address Bits**

Variable	Equation
ADDR_OFFSET	$\log_2$(C_DDR_DWIDTH/8)
COLADDR_STARTBIT	MCI_ADDR_WIDTH – (C_DDR_CAWIDTH+ADDR_OFFSET)
COLADDR_ENDBIT	COLADDR_STARTBIT + C_DDR_CAWIDTH – 1
ROWADDR_STARTBIT	COLADDR_STARTBIT – C_DDR_AWIDTH
ROWADDR_ENDBIT	ROWADDR_STARTBIT + C_DDR_AWIDTH – 1
BANKADDR_STARTBIT	ROWADDR_STARTBIT – C_DDR_BAWIDTH
BANKADDR_ENDBIT	BANKADDR_STARTBIT + C_DDR_BAWIDTH – 1

Table 8 shows an example of the address mapping. In this example, the DDR2 data width is 32, the column address width is 9, the row address width is 13, and the bank address width is 2.

Table 8: **Example Address Mapping (C_DDR_DWIDTH=32, C_DDR_CAWIDTH=9, C_DDR_RAWIDTH=13, C_DDR_BAWIDTH=2)**

Variable	Equation	Value
ADDR_OFFSET	$\log_2$(C_DDR_DWIDTH/8)	$\log_2$(32/8) = 2
COLADDR_STARTBIT	MCI_ADDR_WIDTH – (C_DDR_CAWIDTH+ADDR_OFFSET)	36 – (9 + 2) = 25
COLADDR_ENDBIT	COLADDR_STARTBIT + C_DDR_CAWIDTH – 1	25 + (9 - 1) = 33
ROWADDR_STARTBIT	COLADDR_STARTBIT – C_DDR_AWIDTH	25 – 13 = 12
ROWADDR_ENDBIT	ROWADDR_STARTBIT + C_DDR_AWIDTH – 1	12 + 13 - 1 = 24
BANKADDR_STARTBIT	ROWADDR_STARTBIT – C_DDR_BAWIDTH	12 – 2 = 10
BANKADDR_ENDBIT	BANKADDR_STARTBIT + C_DDR_BAWIDTH – 1	10 + 2 – 1 = 11

Row Conflict and Bank Conflict Mask Settings

The MI_ROWCONFLICT_MASK and MI_BANKCONFLICT_MASK parameters, set by the user, enable the PPC440 Memory Controller Interface (MCI) to detect row address and bank address conflicts between the current and previous command addresses. MI_ROWCONFLICT_MASK and MI_BANKCONFLICT_MASK parameters must be set based on the address offset width, the memory device column address width, the memory device row address width, and the memory device bank address width. The address offset width equals $\log_2$ (C_DDR_DWIDTH/8).

Figure 2 shows the address fields of MIMCADDRESS [0:35], the corresponding 32-bit MI_BANKCONFLICT_MASK, and 32-bit MI_ROWCONFLICT_MASK for a memory interface with the following parameters.

- C_DDR_DWIDTH = 64
- Address Offset = $\log_2$(64/8) = 3
- Column Address Width = 10
- Row Address Width = 14
- Bank Address Width = 3

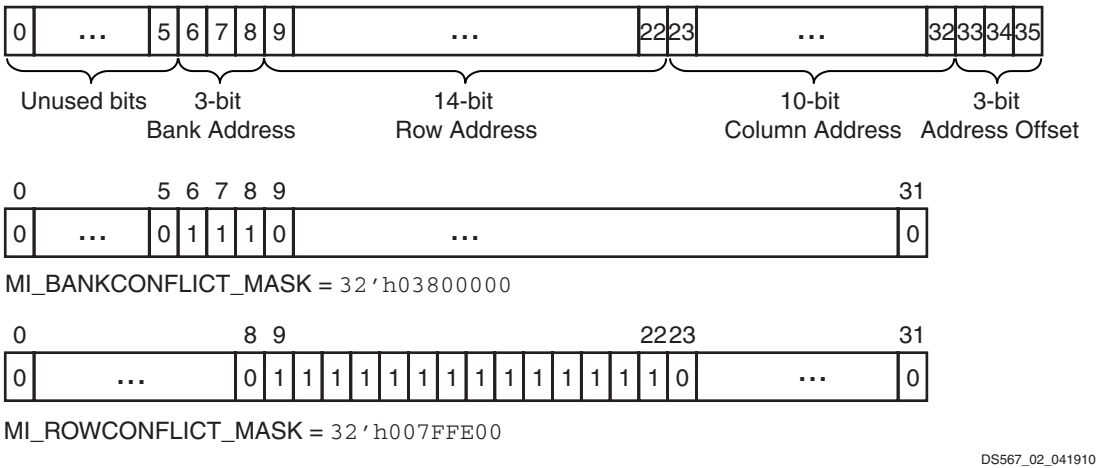

Figure 2: **MIMCADDRESS Bus and Corresponding MI_ROWCONFLICT_MASK and MI_BANKCONFLICT_MASK**

MI_ROWCONFLICT_MASK [0:31] corresponds to MIMCADDRESS [0:31] with the Row Address bits set to ones and all other bits set to zeros to enable conflict detection. And the MI_BANKCONFLICT_MASK [0:31] corresponds to MIMCADDRESS [0:31] with the Bank Address bits set to ones and all other bits set to zeros to enable conflict detection.

Another example with reduced data width:

- C_DDR_DWIDTH = 32
- Address Offset = $\log_2(32/8) = 2$
- Column Address Width = 10
- Row Address Width = 14
- Bank Address Width = 3

The MI_ROWCONFLICT_MASK and MI_BANKCONFLICT_MASK parameters are:

- MI_ROWCONFLICT_MASK = `32'h003fff00`
- MI_BANKCONFLICT_MASK = `32'h01c00000`

Upgrading From PPC440MC_DDR2 v2.00.x to v3

To migrate an existing design from PPC440MC_DDR2v2 to PPC440MC_DDR2v3, the following steps are necessary.

Note: Tool errors are reported if the subsequent steps are not performed.

1. In the MHS file, the C_MEM_DQS_IO_COL, C_MEM_DQ_IO_MS, and C_IDELAYCTRL_LOC parameters are no longer used and must be removed from the PPC440MC_DDR2 instance.
2. In the UCF file, PPC440MC_DDR2 specific constraints containing AREA_GROUP and RLOC_ORIGIN must be removed. Also remove any PPC440MC_DDR2 LOC constraints involving IDELAYCTRL_XxYy sites from the UCF.

Design Implementation

FIFOs

The read and write datapaths have FIFOs for storing data. The FIFOs in Virtex-5 FXT devices have a built-in ECC module. For applications that require ECC, the ECC in the FIFOs can be used. The block RAMs next to the PPC block are used to implement these FIFOs.

There is no latency impact during writes. The PPC440MC DDR2 Memory Controller operates with a write data delay of zero, and the data is in the FIFO when it has to be sent to the memory. For reads, there is a one-cycle latency hit because of using the FIFOs.

Clocking and Reset

The PPC440MC DDR2 Memory Controller requires all four phases of the clock. The CLK180 and CLK270 signals are generated by local inversion using MI_MCCLK and MI_MCCLK90. The MI_MCRESET from the MCI module is synchronized to CLK180, CLK270, MI_MCCLK90, and the MI_MCCLK_200 before being used in the design.

Top-Level Design

The PPC440MC DDR2 Memory Controller is comprised of the controller, the physical layer, and the FIFO interface as shown in Figure 3.

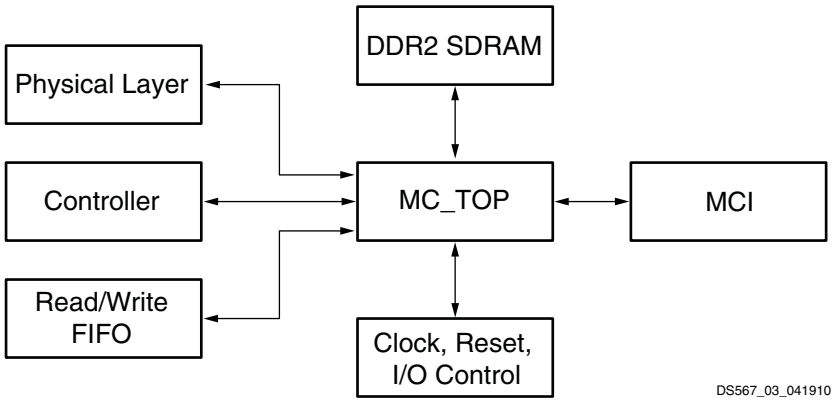

Figure 3: **PPC440MC DDR2 Memory Controller Block Diagram**

Physical Layer

The physical layer is comprised of the write datapath, the read datapath, the calibration state machine for DQS and DQ calibration, the calibration logic for read enable alignment, and the memory initialization state machine. The write datapath generates the data and strobe signals transmitted during a Write command. The read datapath captures the read data in the read strobe domain.

Write Datapath

The write datapath, shown in Figure 4, is implemented using the IOB ODDR in the same edge mode. Two data words are presented on the rising edge of the FPGA clock and the ODDR converts it into DDR data.

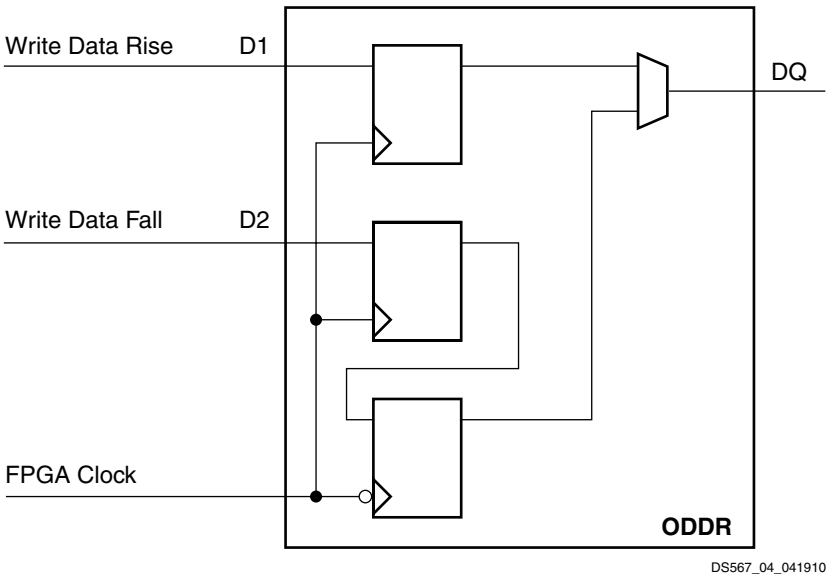

Figure 4: **Write Datapath**

Read Datapath

As outlined in XAPP858 *High-Performance DDR2 SDRAM Interface in Virtex-5 Devices*, the read datapath comprises several flip-flop stages (ranks) over which read data from the DDR2 memory is transferred from the DQS strobe domain to the internal FPGA (CLK0) clock domain. It also includes the local I/O clock network for distributing the DQS for the initial data capture, and circuitry to ensure robust data capture on the last transfer of a read burst:

- Rank 1 Capture: Initial capture of DQ with DQS using the IOB IDDR flip-flop and IDELAY elements. The differential DQS pair must be placed on a clock-capable I/O pair and is distributed through the local BUFIO network to each of the corresponding DQ IDDR capture flip-flops. The IDDR generates two single-data-rate signals at its Q1 and Q2 outputs. Both Q1 and Q2 outputs are clocked by the falling edge of DQS because:
  - The IDDR is configured in SAME_EDGE mode.
  - The DQS is inverted before clocking the IDDR.
- Rank 2 Capture: The IDDR captures the data from the DDR2 memory, which is in the double-data-rate domain, and generates two single-data-rate streams at its Q1 and Q2 outputs. The outputs of the IDDR are transferred to flip-flops that are clocked by the rising or falling edge of CLK0. These flip-flops are located in the CLBs. There are two possible sets of flip-flops that each IDDR output can be transferred to. This allows synchronization of the IDDR outputs to either edge of CLK0 and reduces the maximum number of IDELAY taps required to perform timing alignment.
- Rank 3 Capture: The output of Rank 2 flip-flops clocked by the falling edge of CLK0 are transferred to flip-flops clocked by the rising edge of CLK0. In addition, the multiplexer for each DQ capture circuit is set to choose the appropriate synchronization path used.
- DQS Gate Circuit: This circuit disables the clock enable for each DQ IDDR at the end of a read burst to prevent any glitches associated with the DQS strobe being 3-stated by the memory from clocking the IDDR flip-flop. There is one such circuit per DQS group.

The read datapath is shown in Figure 5.

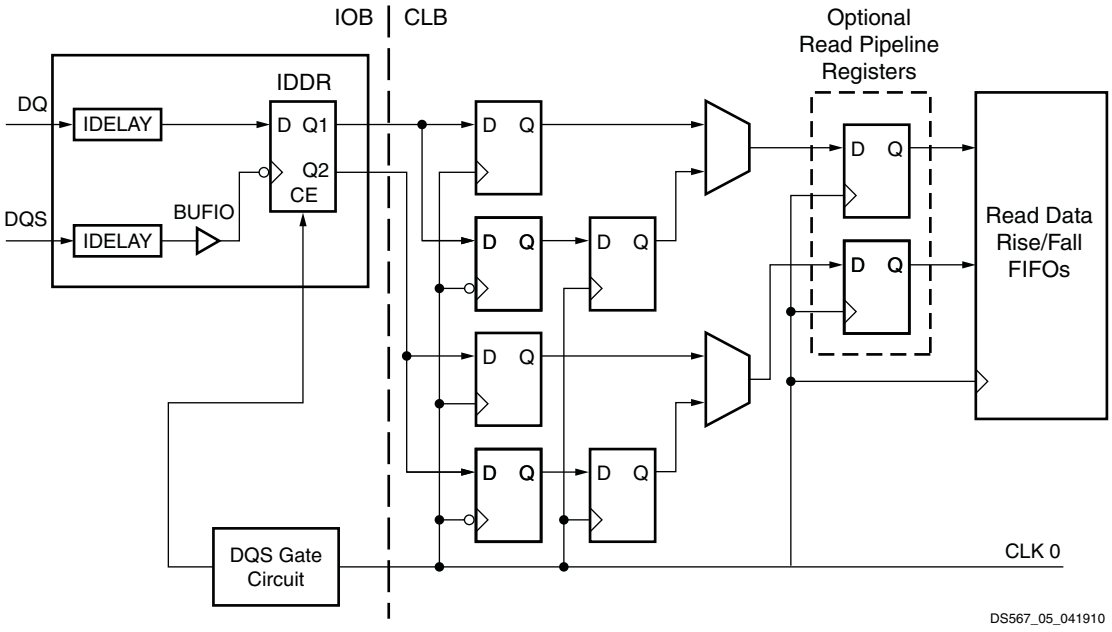

Figure 5: **Read Datapath**

Initialization

The initialization state machine (shown in Figure 6) powers up and initializes the DDR2 device and performs the required reads and writes for the calibration. This state machine provides the required delay and the command sequence required for the DDR2 device. After power up and initialization, the state machine issues the read and write commands that are required for calibration. The state machine asserts an initialization_done signal at the completion of the initialization and calibration. The initialization state machine starts to execute after reset.

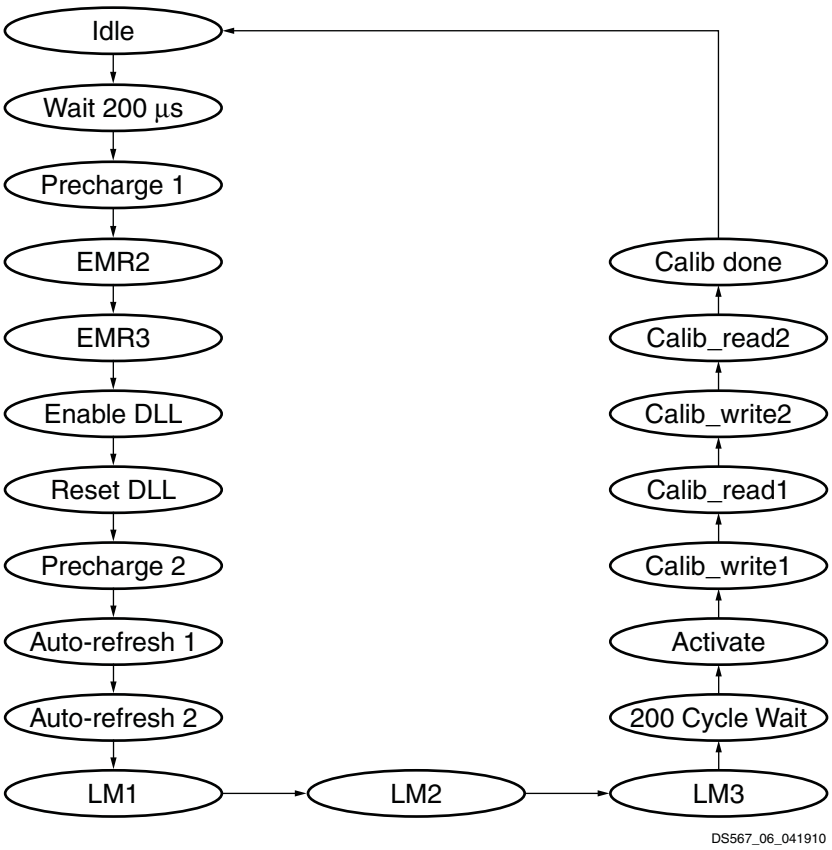

Figure 6: **Initialization State Machine Flow**

Controller

The controller becomes active after the initialization state machine asserts the initialization_done signal. The controller processes the commands from the MCI. The interface between the controller and the MCI is explained in Interfacing to the MCI.

Bank Management

The PPC440MC DDR2 Memory Controller can keep four banks open. The banks are opened as commands are processed by the PPC440MC DDR2 Memory Controller. When four banks are already opened and a command occurs for a different bank, the least recently used bank is closed and the new bank is opened. All the banks are closed during auto refresh. After the auto refresh, the PPC440MC DDR2 Memory Controller opens the bank last accessed before the auto refresh command. The conflict logic in the MCI always compares the current address to the previous address to determine the bank and row conflicts. There are no signals from the PPC440MC DDR2 Memory Controller to the MCI, indicating that all the banks are closed during the auto refresh. The MCI does not assert the conflict bit for the first command that it presents after auto refresh, if the banks and row address of that command

is the same as the command that was last presented before the auto refresh. To avoid missing the conflict, the PPC440MC DDR2 Memory Controller, after the auto refresh command, opens the last accessed bank and row. The flow diagram in Figure 7 shows the bank management flow.

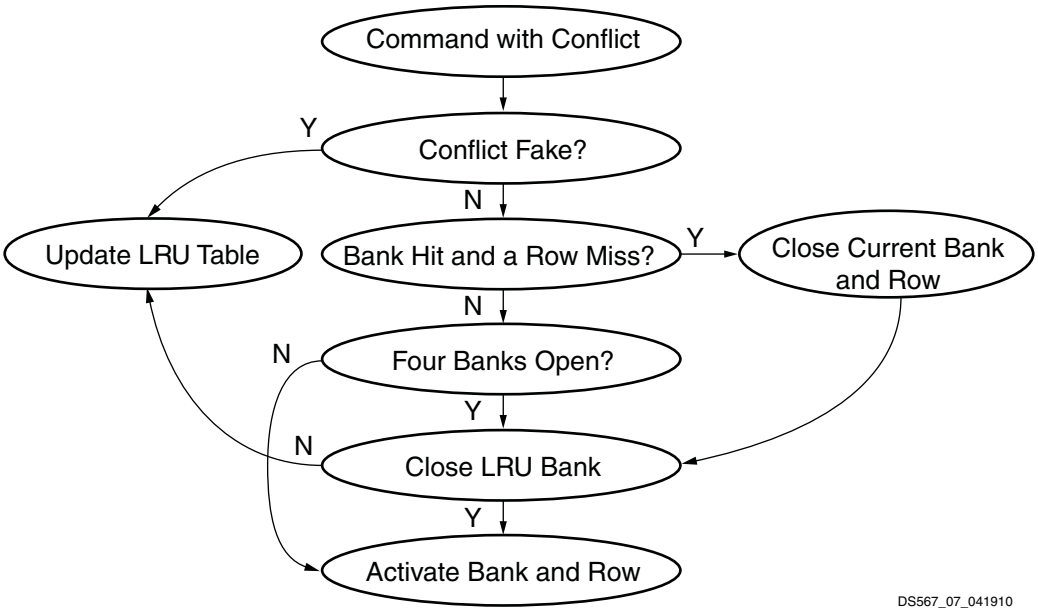

Figure 7: **Bank Management Flow Diagram**

The PPC440MC DDR2 Memory Controller has four internal bank registers. The width of the internal bank registers depends on the row and bank address widths. The internal bank registers hold bank information, row information, and a status bit. The status bit holds the open and close status of the bank and row in the internal bank register. If the status bit is set, the bank and row in the internal register are opened; otherwise the contents of the internal register does not hold any significance. The status bit is cleared during the assertion of reset and auto refresh because the banks and row are closed during that time. The internal bank register contents are updated as bank and rows are opened.

In Figure 8, the internal_bank# waveforms are the internal bank registers. The internal_bank0 register holds the most recently used bank and row, and the internal_bank3 register holds the least recently used bank and row numbers. At clock 1, the PPC440MC DDR2 Memory Controller does not have any banks open. In clock 2, the MCI presents a command for bank 0 and also asserts the conflict bit. The PPC440MC DDR2 Memory Controller opens this bank and performs the write operation. Since no banks are open, the PPC440MC DDR2 Memory Controller does not have to close a bank to open bank 0. The internal_bank0 register has bank 0, row 0 after the active command in clock 4. The MCI presents a write command in clock 20 for bank 1. Because bank 1 has not been opened before, the PPC440MC DDR2 Memory Controller opens bank 1 and performs the write operation. Only one bank, bank 0 was kept open before this command. The PPC440MC DDR2 Memory Controller does not have to close any banks to open bank 1. Bank 1 is now the most recently used bank, internal_bank0 now holds bank 1, row 1, and internal_bank1 holds bank 0, row 0.

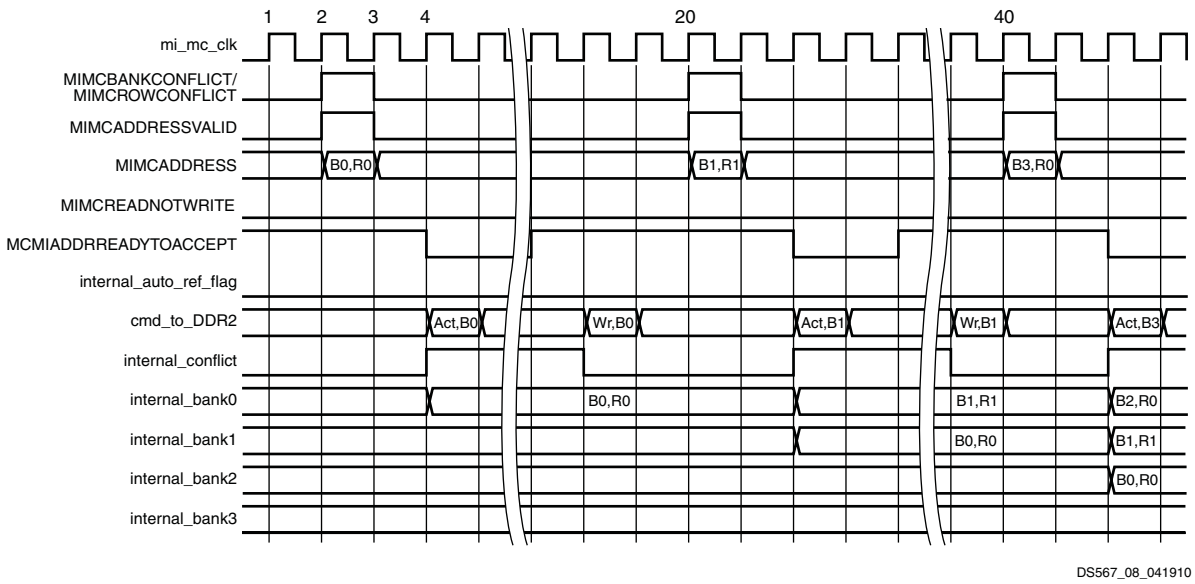

Figure 8: **Bank Management with No Internal Banks Open**

In Figure 9, the PPC440MC DDR2 Memory Controller has four banks open in clock 1. The MCI presents a command for bank 0 in clock 2 and asserts the conflict bit. Bank 0 has already been opened by the PPC440MC DDR2 Memory Controller. The conflict is a fake conflict. The PPC440MC DDR2 Memory Controller does not deassert the MCMIADDRREADYTOACCEPT signal. Internal_bank0 now holds bank 0, indicating it is the most recently used bank. The commands presented in clocks 8, 15, and 22 are also fake conflicts. In clock 29, the MCI presents a command for bank 7. Bank 7 has not been opened by the PPC440MC DDR2 Memory Controller, the conflict is a real conflict. The PPC440MC DDR2 Memory Controller closes the least recently used bank (bank 0, which is held in internal_bank3) and opens bank 7. The PPC440MC DDR2 Memory Controller also deasserts the MCMIADDRREADYTOACCEPT signal to service the conflict.

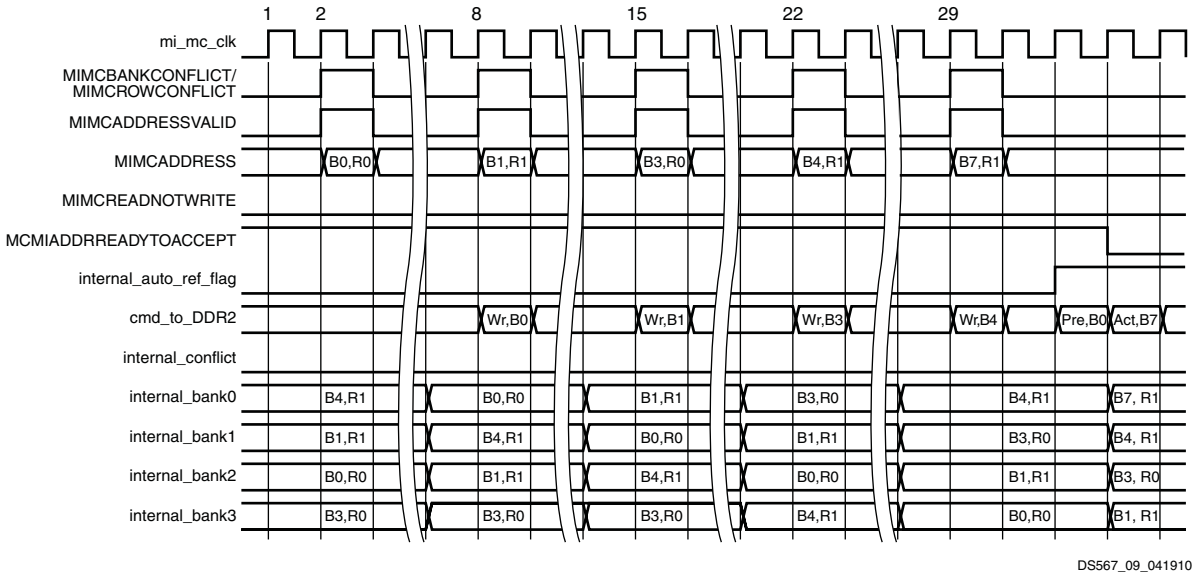

Figure 9: **Bank Management with Four Internal Banks Open**

In Figure 10, the internal auto refresh flag gets asserted in clock 12. The PPC440MC DDR2 Memory Controller closes all the banks to perform the auto refresh function. After the auto refresh command, the PPC440MC DDR2 Memory Controller opens the last accessed bank before the auto refresh command. In this case, the last bank that was accessed before the auto refresh command was bank 7, row 0. The PPC440MC DDR2 Memory Controller opens that bank after the auto refresh command in clock 46. The MCI presets a command in clock 50. The row and the bank are the same as the previous command and the MCI does not assert the conflict bit. The PPC440MC DDR2 Memory Controller has already opened the bank and the row and the command proceeds normally.

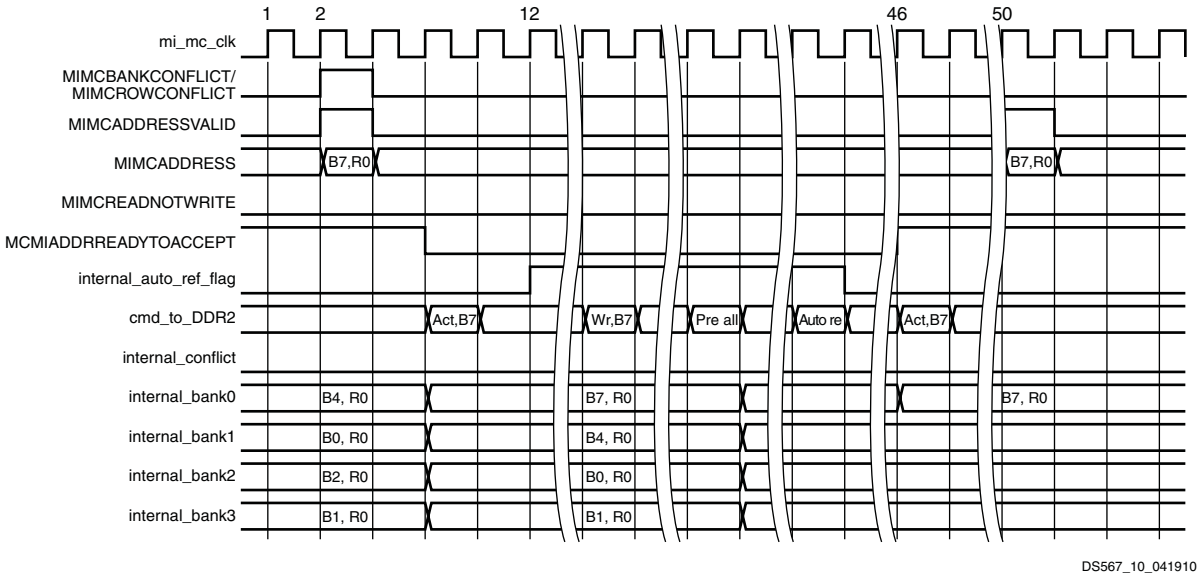

Figure 10: **Bank Management During Auto Refresh**

Interfacing to the MCI

Handshaking between the PPC440MC DDR2 Memory Controller and the MCI is done with the MCMIADDRREADYTOACCEPT signal. The following subsections describe this signal and its operation under various conditions.

The PPC440MC DDR2 Memory Controller asserts the MCMIADDRREADYTOACCEPT signal when it is ready to accept transactions from the MCI. The PPC440MC DDR2 Memory Controller completes all transactions that it had accepted from the MCI. The MCI does not re-request a transaction; the PPC440MC DDR2 Memory Controller must complete all the transactions it accepted. The PPC440MC DDR2 Memory Controller deasserts MCMIADDRREADYTOACCEPT during:

- Auto refresh
- Bank and row conflicts
- Change of direction (write to read and read to write)
- Initialization

Auto Refresh

The PPC440MC DDR2 Memory Controller deasserts the MCMIADDRREADYTOACCEPT signal once the auto refresh flag is asserted internally. The auto refresh flag is asserted by the PPC440MC DDR2 Memory Controller when it has to perform an auto refresh function to the DDR2 memory. Due to the synchronization delay, the MCI sees the deassertion of this signal after a delay of two clock cycles. The MCI can request transactions when the PPC440MC DDR2 Memory Controller deasserts the MCMIADDRREADYTOACCEPT signal because of the synchronization delay. The PPC440MC DDR2 Memory Controller asserts and deasserts this signal two cycles earlier to account for the synchronization delay.

The minimum DDR2 burst size is four. When the PPC440MC DDR2 Memory Controller is set to a burst of 4x64, the MCI must be set to a burst of 2x128. When set to the DDR2 burst size of four, the MCI can request transactions every other clock, which can cause the MCI to request at least one transaction while MCMIADDRREADYTOACCEPT is deasserted. The PPC440MC DDR2 Memory Controller accepts requests from the MCI for up to two clocks after the deassertion of the MCMIADDRREADYTOACCEPT signal. The transaction that was accepted by the PPC440MC DDR2 Memory Controller after the deassertion of the MCMIADDRREADYTOACCEPT signal is processed after the auto refresh command completes.

In Figure 11 through Figure 14, the DDR2 burst size is 4.

In Figure 11, the PPC440MC DDR2 Memory Controller deasserts the MCMIADDRREADYTOACCEPT signal in clock 6 in response to the assertion of the internal auto refresh flag in clock 5. The MCI presents two commands in clock 3 and clock 5, while the MCMIADDRREADYTOACCEPT signal is asserted. The PPC440MC DDR2 Memory Controller processes both commands before the auto refresh command to the memory. The PPC440MC DDR2 Memory Controller asserts the MCMIADDRREADYTOACCEPT signal in clock 15 after the auto refresh command completes.

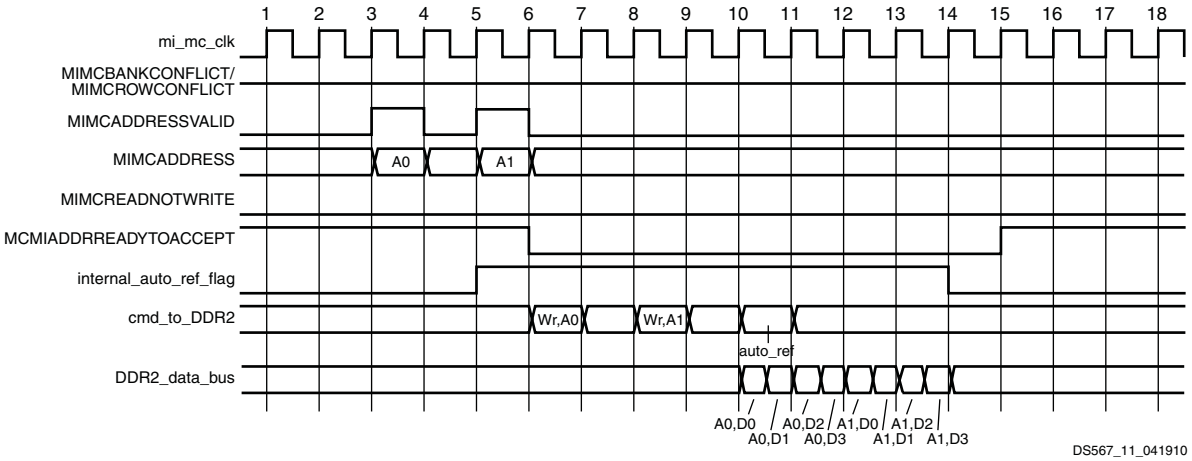

Figure 11: **Write Command During Auto Refresh and MCMIADDRREADYTOACCEPT Asserted**

In Figure 12, the MCI presents a command in clock 7 after the deassertion of the MCMIADDRREADYTOACCEPT signal. The PPC440MC DDR2 Memory Controller processes this command after the auto refresh command completes. The assertion of the MCMIADDRREADYTOACCEPT signal is delayed until the second write command completes. This can be done because the MCI sees the assertion of the MCMIADDRREADYTOACCEPT signal late due to the synchronization delay and has a minimum delay of two PPC440MC DDR2 Memory Controller clocks before presenting a transaction.

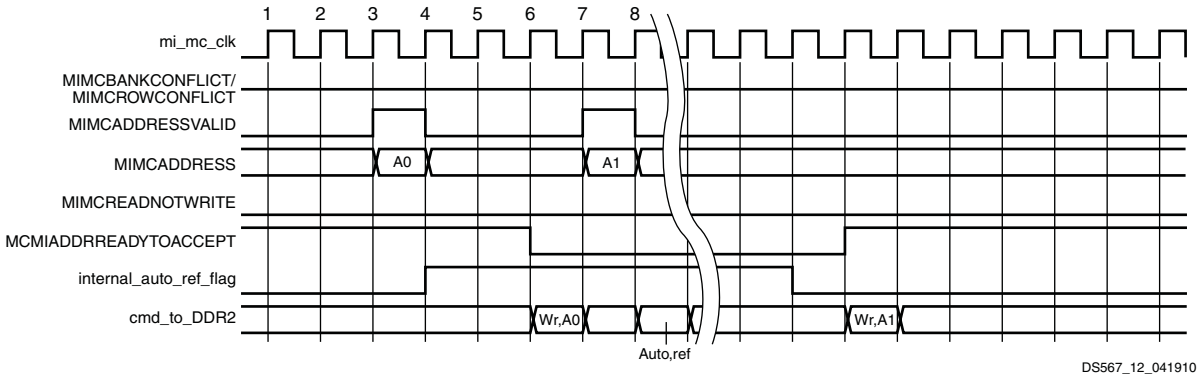

Figure 12: **Write Command During Auto Refresh and MCMIADDRREADYTOACCEPT Deasserted**

Bank and Row Conflicts

During conflicts, the PPC440MC DDR2 Memory Controller deasserts the MCMIADDRREADYTOACCEPT signal to service the conflict. The MCI asserts the conflict signal if the bank or the row in the current access is different from the bank or the row in the previous access. The MCI does not know of all the banks that are kept open in the PPC440MC DDR2 Memory Controller. It keeps track of banks and rows from the current and previous accesses. Because the MCI does not keep track of the history of banks and rows that were opened, the conflict signal asserted by the MCI during some of the access might not be for real conflicts. The PPC440MC DDR2 Memory Controller evaluates the conflict signal from the MCI and determines if the conflict is real or not. During real conflicts, the PPC440MC DDR2 Memory Controller deasserts the MCMIADDRREADYTOACCEPT signal. During fake conflicts, the PPC440MC DDR2 Memory Controller does not deassert MCMIADDRREADYTOACCEPT. When MI_CONTROL:AUTOHOLD is set to 2'b10, the MCI holds off for four PPC440MC DDR2 Memory Controller clocks whenever it asserts the conflict signal.

In Figure 13, the row and bank conflicts are OR'ed together and treated as one conflict. The MCI presents three write commands to the PPC440MC DDR2 Memory Controller. The first write is to bank 0, row 0 (clock 3), the second write is to bank 1, row 1 (clock 5), and the third write is to bank 0, row 0 (clock n+1). The PPC440MC DDR2 Memory Controller already opened bank 0 and row 0 in the previous command (not shown in the figure). The MCI does not assert the conflict for the first write (clock 3) because the bank and row are the same as the bank and row in the previous access.

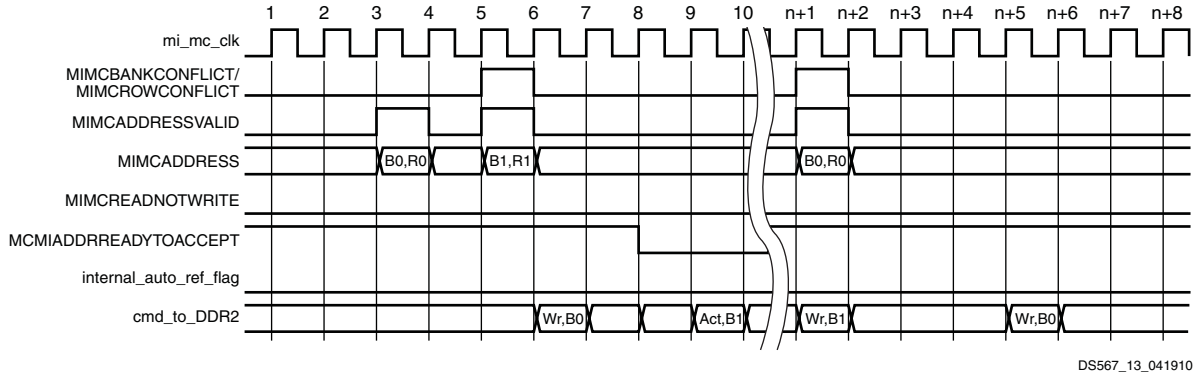

Figure 13: **Read Command with Conflict**

The MCI asserts the conflict bit while presenting the second write (clock 5). The conflict bit is asserted because the row and bank are different from the row and bank in write 1 (clock 3). The PPC440MC DDR2 Memory Controller had not opened bank 1, row 1 during any of its previous accesses. The PPC440MC DDR2 Memory Controller evaluates this conflict as a real conflict and deasserts the MCMIADDRREADYTOACCEPT signal in two or more clock cycles. The PPC440MC DDR2 Memory Controller opens bank 1, row 1 and issues the write command. The PPC440MC DDR2 Memory Controller asserts the MCMIADDRREADYTOACCEPT signal when it is ready to accept new requests.

When presenting the third write in clock n+1, the MCI asserts the conflict bit. It asserts the conflict bit because the bank and row in the third write (clock n+1) is different from the bank and row in the second write (clock 7). The PPC440MC DDR2 Memory Controller does not treat the third write (clock n+1) as a conflict because the bank and row for the third write were opened during the first write and have not been closed. The PPC440MC DDR2 Memory Controller evaluates this conflict as a fake conflict and does not deassert the MCMIADDRREADYTOACCEPT signal. The PPC440MC DDR2 Memory Controller takes at least two clock cycles to evaluate the conflict, and there are at least four idle cycles in the DDR2_data_bus.

Change of Direction

The PPC440MC DDR2 Memory Controller deasserts MCMIADDRREADYTOACCEPT during write_to_read or read_to_write to meet the DDR2 specification on change of direction. The duration of the deassertion depends on the DDR2 specification. The MCI does not present any commands during the deassertion of MCMIADDRREADYTOACCEPT in this case because it automatically holds off for four clocks, and it also sees the MCMIADDRREADYTOACCEPT deassertion during the auto hold off duration.

In Figure 14, the MCI presents a write command in clock 3 followed by a read command in clock 5. In response to the read command in clock 5, the PPC440MC DDR2 Memory Controller deasserts MCMIADDRREADYTOACCEPT and keeps it deasserted until the write_to_read turnaround time is met. The write_to_read turnaround and read_to_write turnaround times depend on the memory vendor specification. In clock 11 the MCI presents a write command. The PPC440MC DDR2 Memory Controller does not deassert MCMIADDRREADYTOACCEPT because the read_to_write turnaround time is two clocks. It is covered by the auto hold off of four clocks by the MCI.

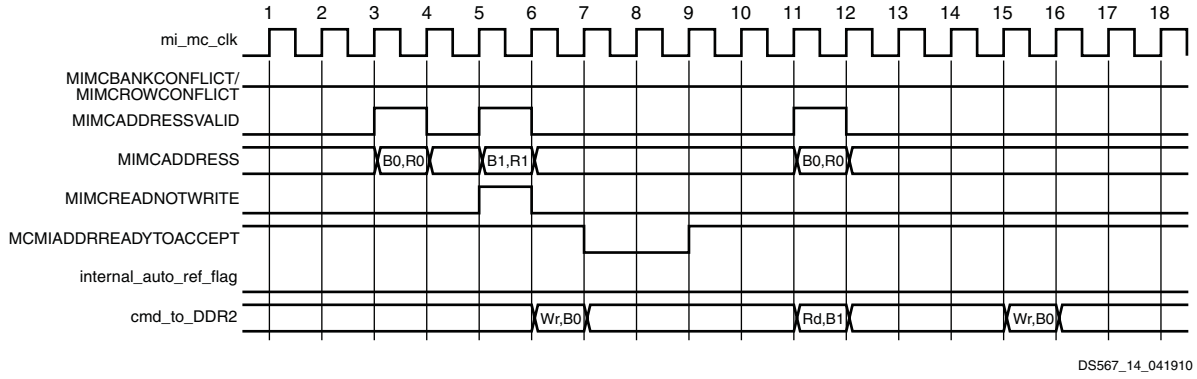

Figure 14: **Write Command Followed by Read Command**

Initialization

The MCMIADDRREADYTOACCEPT signal is deasserted during the entire duration of initialization and calibration. At the end of the initialization and calibration sequence, the PPC440MC DDR2 Memory Controller asserts the MCMIADDRREADYTOACCEPT signal. The calibration sequence is done once during initialization and is not repeated.

ECC Feature

The built in Hamming code error correction feature in the Virtex-5 FXT FPGA block RAM using the FIFO36_72 primitive supports the ECC feature in this memory controller. With this feature, double bit errors can be detected and a single bit error can be corrected. The MCMIREADDATAERR signal provides the double bit error status and should only be considered when the C_INCLUDE_ECC_SUPPORT parameter is asserted. Single bit error status can be obtained from the rd_ecc_err[1] signal in the mem_if_top.v module. The rd_ecc_err[1] signal is not provided to the MCI interface. The ECC feature is only supported with 128-bit MCI data width and 72-bit external memory interface width.

Bit 16 of the PPC440 MI_CONTROL register or the C_PPC440MC_CONTROL MHS parameter must be set to '1' when the ECC feature in the memory controller is enabled with the C_INCLUDE_ECC_SUPPORT parameter.

Read Modify Write

The computed ECC value for a write with partial byte enables will be incorrect because the disabled bytes stored in the external memory location being accessed are not available during ECC calculation. Therefore, in case of writes with partial byte enables the controller must issue an additional read to re-compute the ECC and write it back to the external memory device. When the memory controller detects a write with partial byte enables, the MCMIADDRREADYTOACCEPT signal is deasserted to prevent the MCI from issuing any more read or write command requests. The MCMIADDRREADYTOACCEPT signal remains deasserted until the additional read command (RMW_READ) followed by a write (RMW_WRITE) command have been executed by the memory controller. The data read back by the RMW_READ command is modified with data to be written for enabled bytes and stored in block RAM FIFOs (FIFO36_72) with the ECC feature turned ON. The RMW_WRITE command simply reads the data to be written out of these FIFOs with the correct ECC value. At the end of the RMW_WRITE transaction the MCMIADDRREADYTOACCEPT signal is asserted.

The target technology is an FPGA listed in the Supported Device Family field of the LogiCORE IP Facts table.

Device Utilization and Performance Benchmarks

The HDL implementation modules automatically instantiate the necessary FPGA OBUF and IOBUF resources for the DDR I/O signals.

To analyze the timing within the FPGA, the design has been implemented to illustrate the FPGA performance and resource utilization values as shown in Table 9.

Table 9: **PPC440MC_DDR2 FPGA Performance and Resource Utilization for XC5VFX100T-FF1136**

Speed Grade	f_{MAX} (MHz)	Parameter Values				Device Resources			
		C_DDR_DWIDTH	C_NUM_RANKS_MEM	C_INCLUDE_ECC_SUPPORT	C_MIB_CM_CLOCK_RATIO	Slices	Block RAM	Slice Flip-Flops	6-input LUTs
-1	267	16-bit	1	0	1:1	540	3	1105	1019
-2	300								
-3	333								
-1	267	32-bit	1	0	1:1	680	3	1415	1255
-2	300								
-3	333								
-1	267	64-bit	1	0	1:1	900	4	2025	1985
-2	300								
-3	333								
-1	220	72-bit	1	1	1:1	1527	6	2942	2230
-2	240								
-3	270								

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/15/08	1.0	Initial Xilinx release with ISE® 10.1 EDK software release.
03/19/08	1.1	Release with ISE 10.1 SP1 EDK software release.
03/31/08	1.1.1	Minor publication issue.
05/09/08	1.2	Revised version of reference design to v2_00_a. In Table 2, removed C_MIBDATA_WIDTH and added default values in C_MEM_BASEADDR, C_MEM_HIGHADDR, C_REG_DIMM, C_DQS_BITS, C_DQ_BITS, and C_NUM_IDELAYCTRL. Changed frequency for 72-bit C_DDR_DWIDTH in Table 9.
8/13/08	1.3	Incorporated CRs 477715 and 477717 to correct signal names and descriptions in Table 1.
10/15/08	1.4	Incorporated CR 479100; corrected Variable values in Table 8.
01/09/09	1.5	Incorporated edits to fix CRs 491863, 491981, 493334, 495692, and 496213.
04/24/09	1.6	Replaced references to supported device families and tool name(s) with hyperlinks to PDF files; Updated trademark information.
9/16/09	1.7	Updated for EDK_L 11.3 release; created v2.00c; added Memory Part Parameter Considerations section; updated images.
10/19/09	1.8	Updated for EDK_L 11.4 release; created v3.00a; added IDELAYCTRL Shared Pinout Considerations and Upgrading From PPC440MC_DDR2 v2.00.x to v3 sections.
4/19/10	1.9	Updated for 12.1 release.
3/11/11	2.0	Created v3.00c; incorporated CR566505; updated for the 13.1 release.

Notice of Disclaimer