# DDR2 Memory Controller for PowerPC 440 Processors

DS567 (v1.2) April 23, 2008

## Introduction

This data sheet describes the DDR2 Memory Controller reference design for the PowerPC® 440 block embedded in the Virtex™-5 FXT Platform FPGAs. It 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

| Reference Design Facts | | |
|---|---|---|
| **Reference Design Specifics** | | |
| Supported Device Family | Virtex-5 FXT Platform FPGAs | |
| Version of Reference Design | PPC440MC | v1_02_a |
| Resources Used | | |
| LUTs | See Table 9 | |
| FFs | See Table 9 | |
| Block RAMs | See Table 9 | |
| Special Features | None | |
| **Provided with Reference Design** | | |
| Documentation | Product Specification | |
| Design File Formats | Verilog | |
| Constraints File | UCF – in EDK PCORE directory | |
| Verification | Verilog Testbench | |
| Instantiation Template | Verilog Wrapper | |
| **Design Tool Requirements** | | |
| Xilinx Implementation Tools | ISE™ 10.1 SP1 or later | |
| Verification | ModelSim SE/EE 6.0c or later | |
| Simulation | ModelSim SE/EE 6.0c or later | |
| Synthesis | XST | |
| **Support** | | |
| See "Notice of Disclaimer." | | |

## 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, users should use the relevant optimal UCF provided in the EDK PCORE directory. There is only one optimal UCF for each device/package/processor combination.

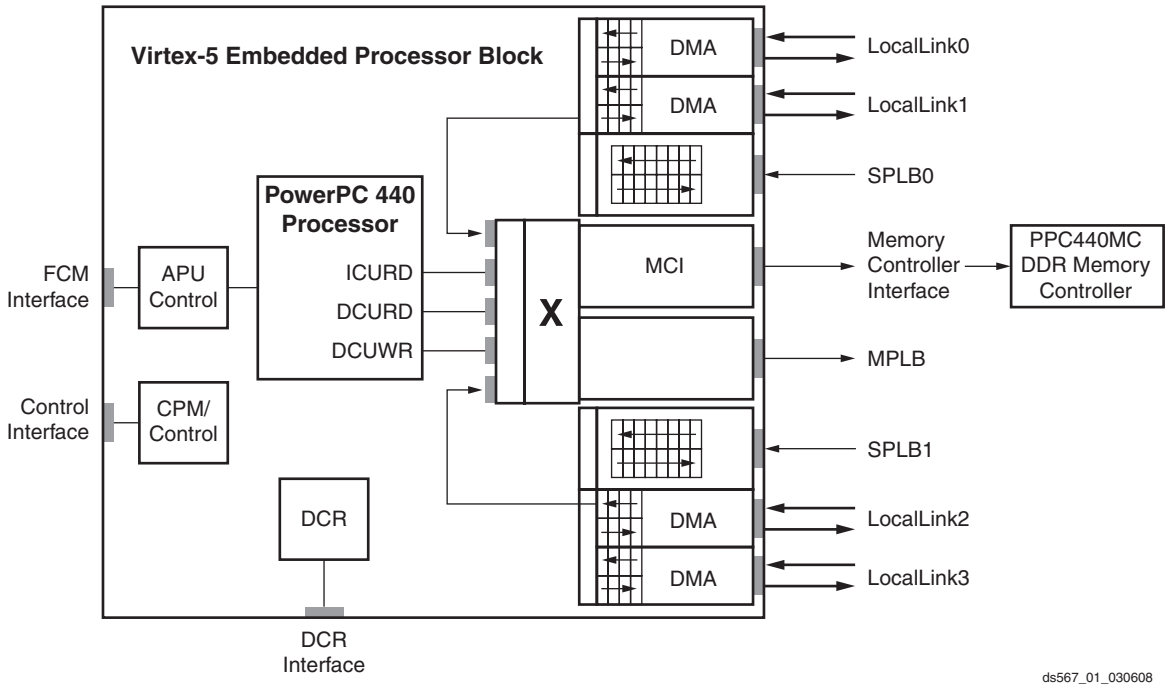


*Figure 1:* **PPC440 MCI and PPC440MC DDR2 Memory Controller Block Diagram**

## I/O Signals

Table 1 defines the PPC440MC DDR2 Memory Controller signals.

*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_CAWIDTH – 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** | | | | |
| MI_MCCLK | CLK | I | | Clock |
| MI_MCCLK90 | CLK | I | | Clock phase shifted by 90 |
| MI_MCCLKDIV2 | CLK | I | | Clock divided by 2 |
| MI_MCCLK_200 | CLK | I | | IDELAY reference clock |
| MI_MCRST | RST | I | | Reset |

## 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** | | | |
| Target FPGA family | C_FAMILY | | |
| Base Address for Memory | C_MEM_BASEADDR | | |
| High Address for Memory | C_MEM_HIGHADDR | | |
| CPMINTERCONNECTCLK to PPC440MC DDR2 clock ratio | C_MIB_MC_CLOCK_RATIO | 0 or 1 for 1:1, 2:1, or 3:1clock ratios<br>2 for 3:2 clock ratio | 0 |
| DDR2 clock period ($t_{CK}$) in ps | C_MC_MIBCLK_PERIOD_PS | 3000 to 8000 | 3000 |
| MCI data width | C_MIBDATA_WIDTH | 128 | 128 |
| Number of generated clock pairs supplied to the DDR2 memory | C_NUM_CLK_PAIRS | 1, 2, 4 | 1 |
| Supported number of external DDR2 memory banks | C_NUM_RANKS_MEM | 1, 2, 4 | 1 |
| Include ECC Logic | C_INCLUDE_ECC_SUPPORT | 0 = Disable<br>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<br>1 = Registered memory | 1 |
| On Die Termination Selection | C_DDR2_ODT_SETTING | 0 = Disables ODT<br>1 = ODT enabled, $R_{TT}$ = 75Ω<br>2 = ODT enabled, $R_{TT}$ = 150Ω<br>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 |

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

| Feature/Description | Parameter Name | Allowable Values | Default Value |
|---|---|---|---|
| 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 | — | |
| $\log_2$ of C_DDR_DWIDTH | C_DQ_BITS | — | |
| $\log_2$ of C_NUM_RANKS_MEM | C_CS_BITS | — | 0 |
| Optional pipeline stage in read data path | C_READ_DATA_PIPELINE | 0 or 1 | 0 |
| I/O column location of DQS groups | C_DQS_IO_COL(4)(5) | See Table 3 | |
| Master/Slave location of DQ I/O | C_DQ_IO_MS(5) | See Table 3 | |
| Number of IDELAYCTRLs required ($\log_2$ of C_DQS_WIDTH) | C_NUM_IDELAYCTRL(6) | 1, 2, 3 | |

**Notes:**

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
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
4. C_DQS_IO_COL is always be 16'd0 because the PPC440MC interfaces use the left column banks.
5. If the optimal UCF provided in the pcore directory is not used and DQS and DQ bits are not placed in the banks recommended in the optimal UCF then a PERL script (ftp://ftp.xilinx.com/pub/applications/misc/ar29313.zip) must be used to determine the correct values of C_DQS_IO_COL and C_DQ_IO_MS. This script requires the user UCF as its input. The PERL script outputs a UCF and a text file. The contents of the output UCF with additional constraints for DQS and DQ must be appended to the user UCF. The text file with the correct values of C_DQS_IO_COL and C_DQ_IO_MS must be copied to the .mhs file. The value for C_DQ_IO_MS depends on DQ pin allocation.
6. The C_NUM_IDELAYCTRL parameter affects the number of instantiations of the IDELAYCTRL primitive in the RTL code. This parameter value has to be set to 1, 2, or 3 depending on the number of FPGA banks used for DQS/DQ signals. For example, if there are three FPGA banks being used to place DQS/DQ signals for a 64-bit memory interface then this parameter C_NUM_IDELAYCTRL equals 3. The location constraints of the IDELAYCTRL primitives have to be set in the UCF. The location coordinates for the IDELAYCTRL primitive depend on the FPGA bank being used and can be determined using FPGA editor. The optimal UCFs provided in the pcore directory already have the location constraints for the IDELAYCTRL primitives.

*Table 3:* **C_DQS_IO_COL and C_DQ_IO_MS Parameter Values for .mhs File**

| UCF Name | C_DQS_IO_COL | C_DQ_IO_MS |
|---|---|---|
| ML507 | 0b0000000000000000 | 0b11101010011110100001111000111100010111011000011110000011011110 0 |
| ddr2_16_fx30t_ff665_banks_11_13 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx70t_ff665_banks_11_13 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx70t_ff1136_banks_11_13 | 0b0000 | 0b1010101010101011 |
| ddr2_16_fx100t_ff1136_banks_17_21 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx100t_ff1136_banks_19_15 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx100t_ff11738_banks_17_21 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx100t_ff11738_banks_19_15 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx130t_ff11738_banks_17_21 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx130t_ff11738_banks_19_15 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx200t_ff11738_banks_17_21 | 0b0000 | 0b1010101011010101 |
| ddr2_16_fx200t_ff11738_banks_19_15 | 0b0000 | 0b1010101011010101 |
| ddr2_32_fx30t_ff665_banks_11_13_17 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx70t_ff665_banks_11_13_17 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx70t_ff1136_banks_11_13_17 | 0b00000000 | 0b10101010101010110101010101010101 |
| ddr2_32_fx100t_ff1136_banks_17_21_25 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx100t_ff1136_banks_19_15_11 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx100t_ff11738_banks_17_21_25 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx100t_ff11738_banks_19_15_11 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx130t_ff11738_banks_17_21_25 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx130t_ff11738_banks_19_15_11 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx200t_ff11738_banks_17_21_25 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_32_fx200t_ff11738_banks_19_15_11 | 0b00000000 | 0b10101010110101010101101010101010 |
| ddr2_64_fx30t_ff665_banks_15_11_13_17 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx70t_ff665_banks_15_11_13_17 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx70t_ff1136_banks_15_11_13_17 | 0b0000000000000000 | 0b1101010110101010101010101010110101 0101010101010111010101101010 |
| ddr2_64_fx100t_ff1136_banks_13_17_21_25 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx100t_ff1136_banks_23_19_15_11 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx100t_ff11738_banks_13_17_21_25 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx100t_ff11738_banks_23_19_15_11 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx130t_ff11738_banks_13_17_21_25 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx130t_ff11738_banks_23_19_15_11 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx200t_ff11738_banks_13_17_21_25 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |
| ddr2_64_fx200t_ff11738_banks_23_19_15_11 | 0b0000000000000000 | 0b0101011010101010101010101101010101 0110101010101011101010101011010 |

*Table 3:* **C_DQS_IO_COL and C_DQ_IO_MS Parameter Values for .mhs File**

| UCF Name | C_DQS_IO_COL | C_DQ_IO_MS |
|---|---|---|
| ddr2_72_fx30t_ff665_banks_15_11_13_17 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx70t_ff665_banks_15_11_13_17 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx70t_ff1136_banks_15_11_13_17 | 0b00000000000000000 | 0b10110101010101101010101010101010101010110101010101010111010101101010 |
| ddr2_72_fx100t_ff1136_banks_13_17_21_25 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx100t_ff1136_banks_23_19_15_11 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx100t_ff11738_banks_13_17_21_25 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx100t_ff11738_banks_23_19_15_11 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx130t_ff11738_banks_13_17_21_25 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx130t_ff11738_banks_23_19_15_11 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx200t_ff11738_banks_13_17_21_25 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |
| ddr2_72_fx200t_ff11738_banks_23_19_15_11 | 0b00000000000000000 | 0b10101101010101101010101010101010110101010101101010101010110101010101101010101010110 |

## Allowable Parameter Combinations

The PPC440MC DDR2 Memory Controller allows up to four external ranks 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.

The following is an example of an acceptable setting for four external ranks (for example, separate DIMMs) of addressable memory, each equal in size to 256 MB. All addressable memory spaces are accessible by all port modules.

```
C_NUM_MEM_RANKS = 4
C_MEM_BASEADDR=0x0000_0000
C_MEM_HIGHADDR=0x3FFF_FFFF
Equates to:
Memory Rank 0 Base Address = 0x0000_0000
Memory Rank 0 High Address = 0x0FFF_FFFF
Memory Rank 1 Base Address = 0x1000_0000
Memory Rank 1 High Address = 0x1FFF_FFFF
Memory Rank 2 Base Address = 0x2000_0000
Memory Rank 2 High Address = 0x2FFF_FFFF
Memory Rank 3 Base Address = 0x3000_0000
Memory Rank 3 High Address = 0x3FFF_FFFF
```

## Parameter - Port Dependencies

Table 4 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 4:* **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 | C_MEM_TYPE | 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 5 shows the interconnection from the MCI to the PPC440MC DDR2 Memory Controller and from the PPC440MC DDR2 Memory Controller to the DDR2 memory.

*Table 5:* **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]<br>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] |

## 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) | 32 – (9 + 2) = 21 |
| COLADDR_ENDBIT | COLADDR_STARTBIT + C_DDR_CAWIDTH – 1 | 21 + (9 - 1) = 29 |
| ROWADDR_STARTBIT | COLADDR_STARTBIT – C_DDR_AWIDTH | 21 – 13 = 8 |
| ROWADDR_ENDBIT | ROWADDR_STARTBIT + C_DDR_AWIDTH – 1 | 8 + 13 - 1 = 20 |
| BANKADDR_STARTBIT | ROWADDR_STARTBIT – C_DDR_BAWIDTH | 8 – 2 = 6 |
| BANKADDR_ENDBIT | BANKADDR_STARTBIT + C_DDR_BAWIDTH – 1 | 6 + 2 – 1 = 7 |

## 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

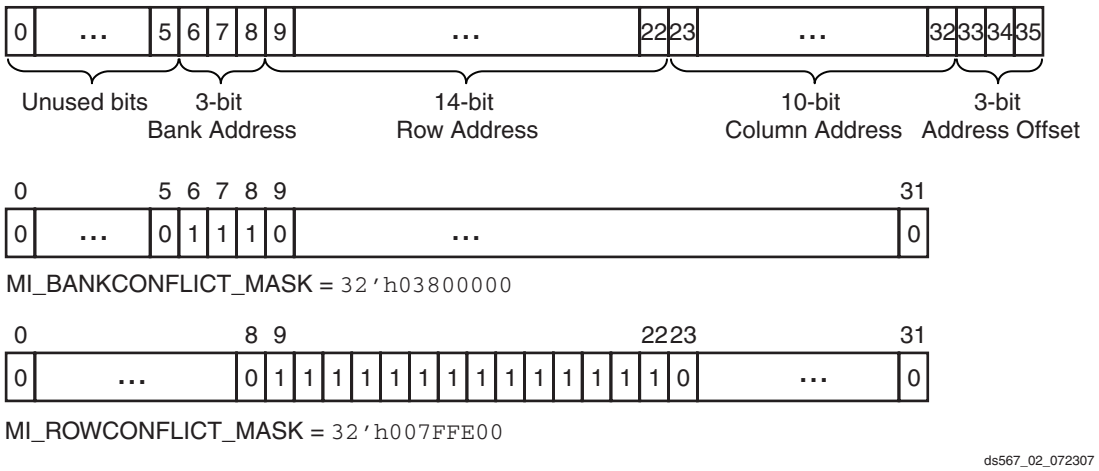


*Figure 2:* **MIMCADDRESS Bus and Corresponding MI_ROWCONFLICT_MASK and MI_BANKCONFLICT_MASK**

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`

# Design Implementation

## 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.

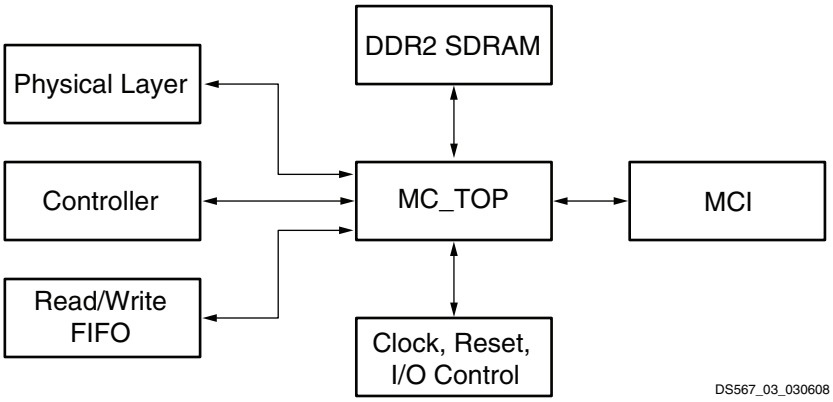


*Figure 3:* **PPC440MC DDR2 Memory Controller Block Diagram**

## 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.

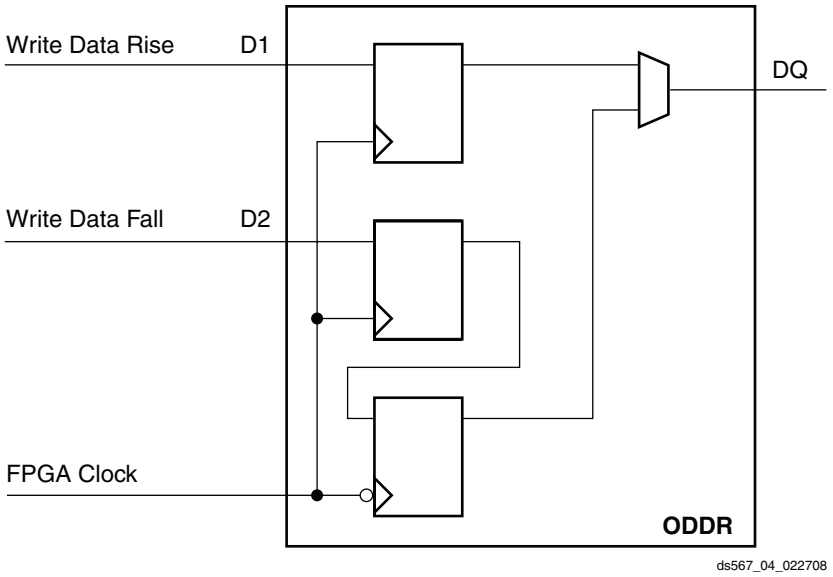


*Figure 4:* **Write Datapath**

### 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.

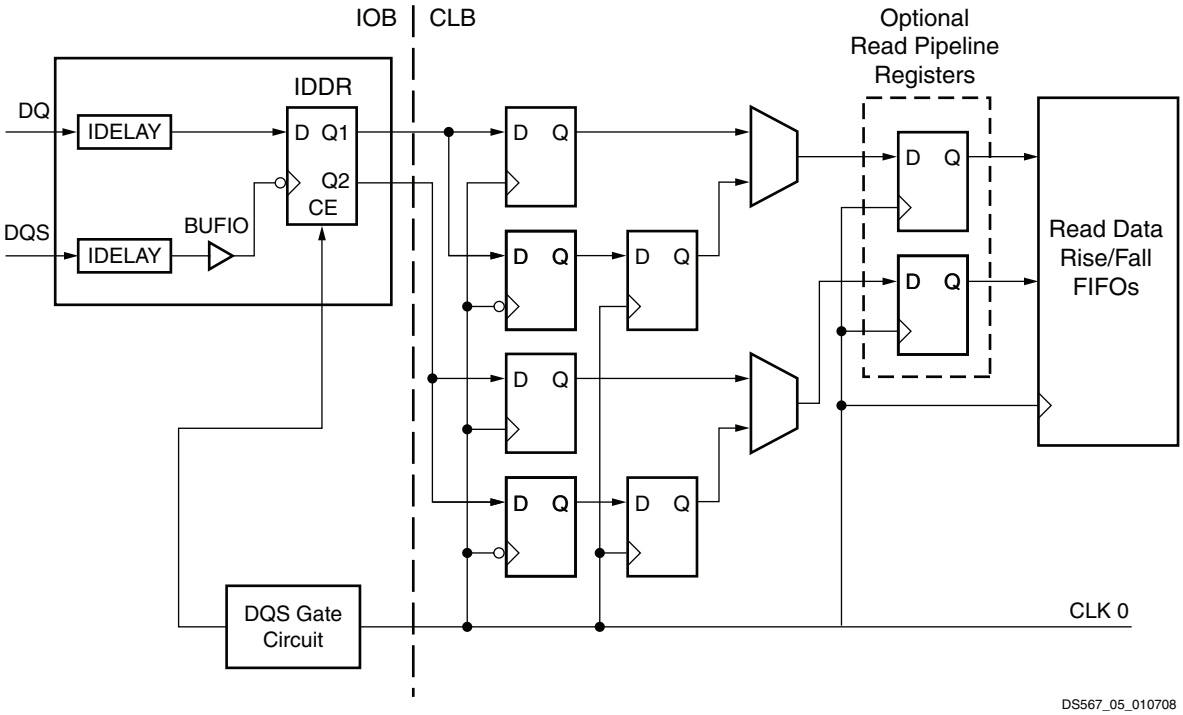


*Figure 5:* **Read Datapath**

## 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.

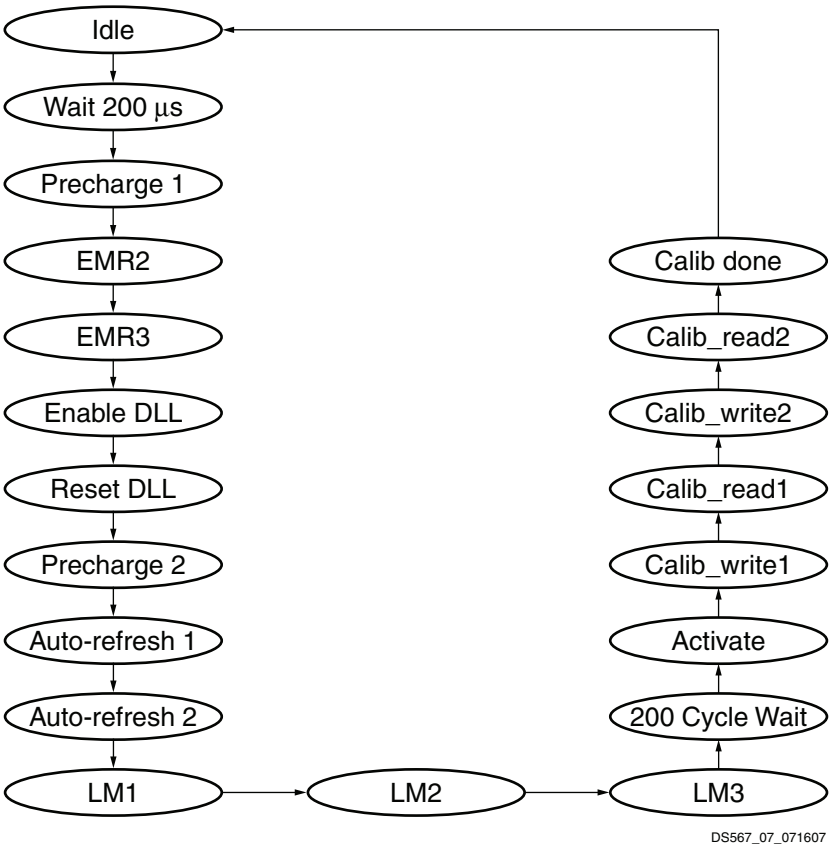


*Figure 6:* **Initialization State Machine Flow**

## 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."

## FIFOs

The read and write datapaths have FIFOs for storing data. The FIFOs in Virtex-5 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 clocks and the reset signal are provided by the MCI. The MCI provides MI_MCCLK (at interface frequency), MI_MCCLK90 (at interface frequency), MI_MCCLK_200 (200 MHz), and MI_MCRST synchronized to MI_MCCLK. 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_MCRST from the MCI module is synchronized to CLK180, CLK270, MI_MCCLK90, and the MI_MCCLK_200 before being used in the design.

## 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.

### MCMIADDRREADYTOACCEPT

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 has to 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 7 through Figure 14, the DDR2 burst size is 4.

In Figure 7, 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.

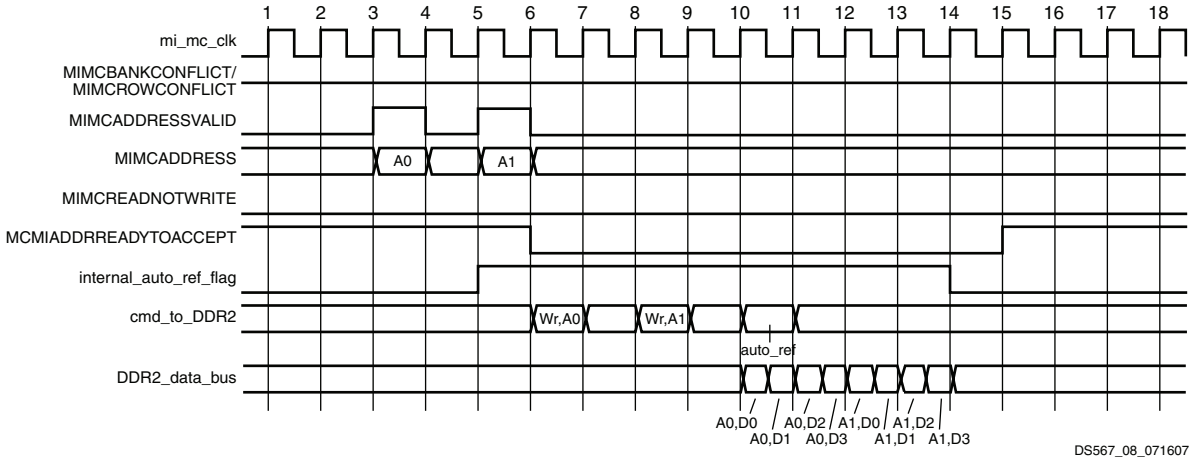


*Figure 7:* **Write Command During Auto Refresh and MCMIADDRREADYTOACCEPT Asserted**

In Figure 8, 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.

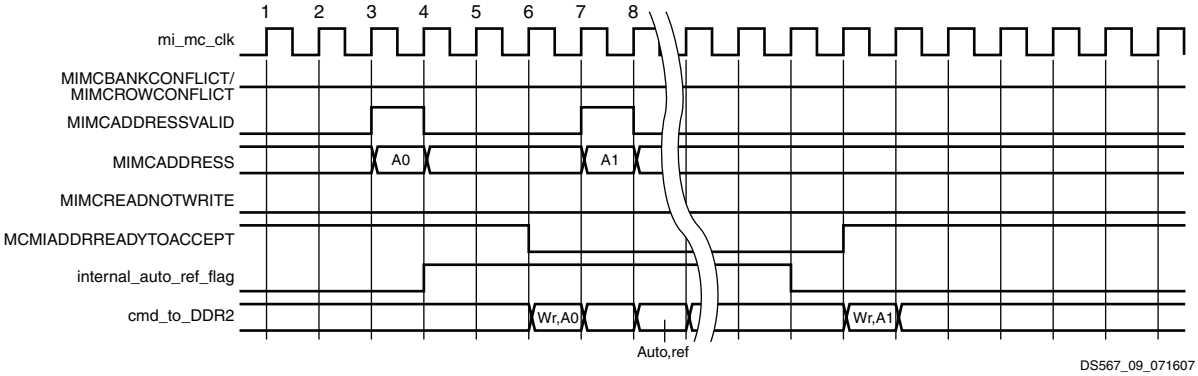


*Figure 8:* **Write Command During Auto Refresh and MCMIADDRREADYTOACCEPT Deasserted**

### 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 9, the row and bank conflicts are ORed 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.

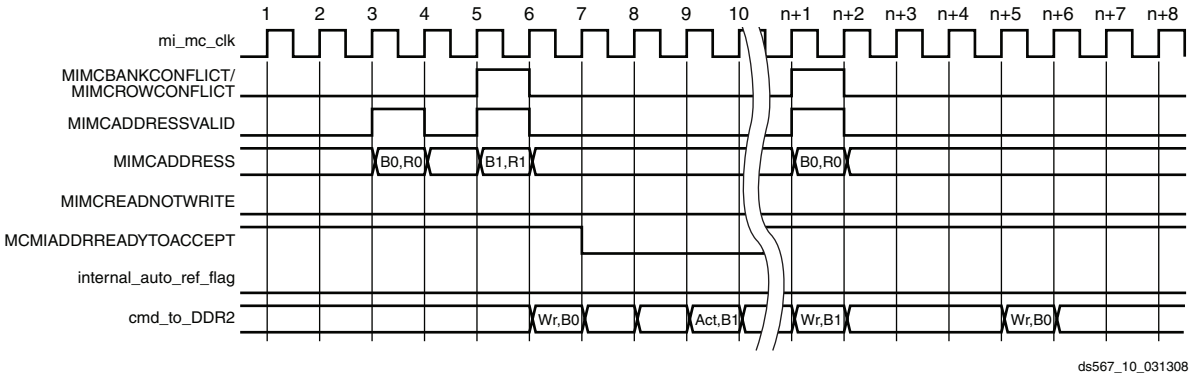


*Figure 9:* **Read Command with Conflict**

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 clock 7. The PPC440MC DDR2 Memory Controller opens bank 1, row 1 and issues the write command. The PPC440MC DDR2 Memory Controller asserts the MCMIADDRREADYTOACCEPT signal in clock 9 after 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 one clock 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 10, 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.

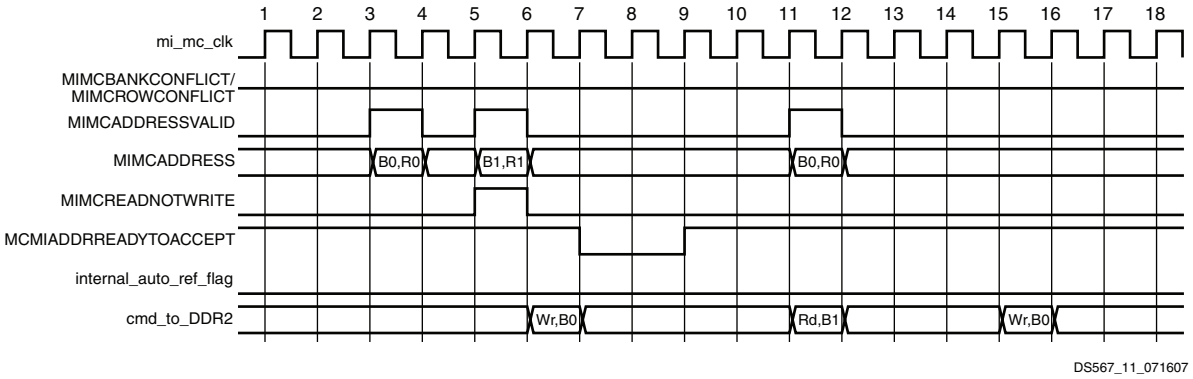


*Figure 10:* **Write Command Followed by Read Command**

### 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.

## 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 11 shows the bank management flow.

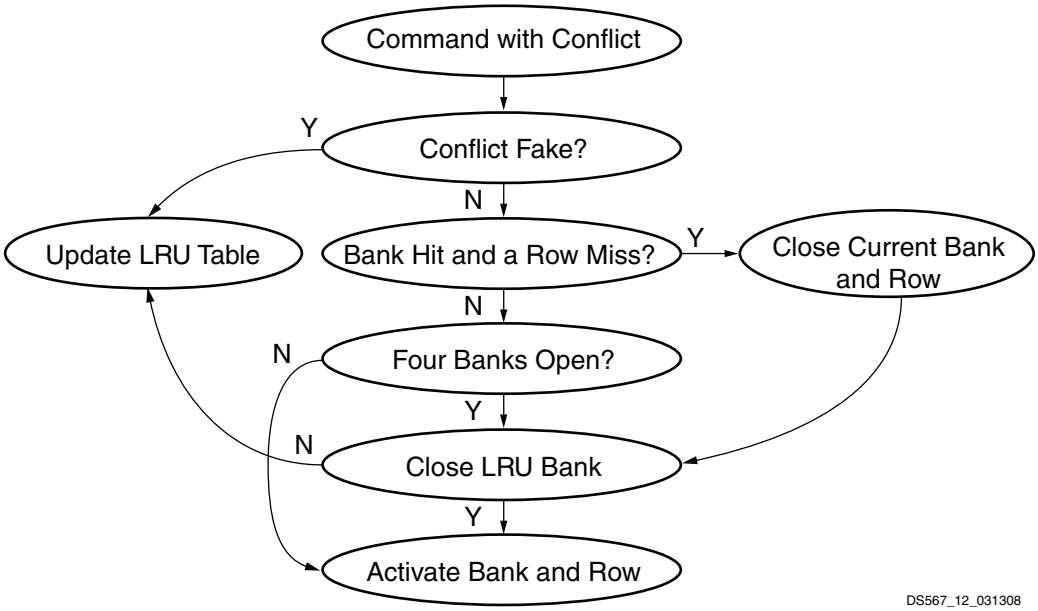


*Figure 11:* **Bank Management Flow Diagram**

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 12, 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.

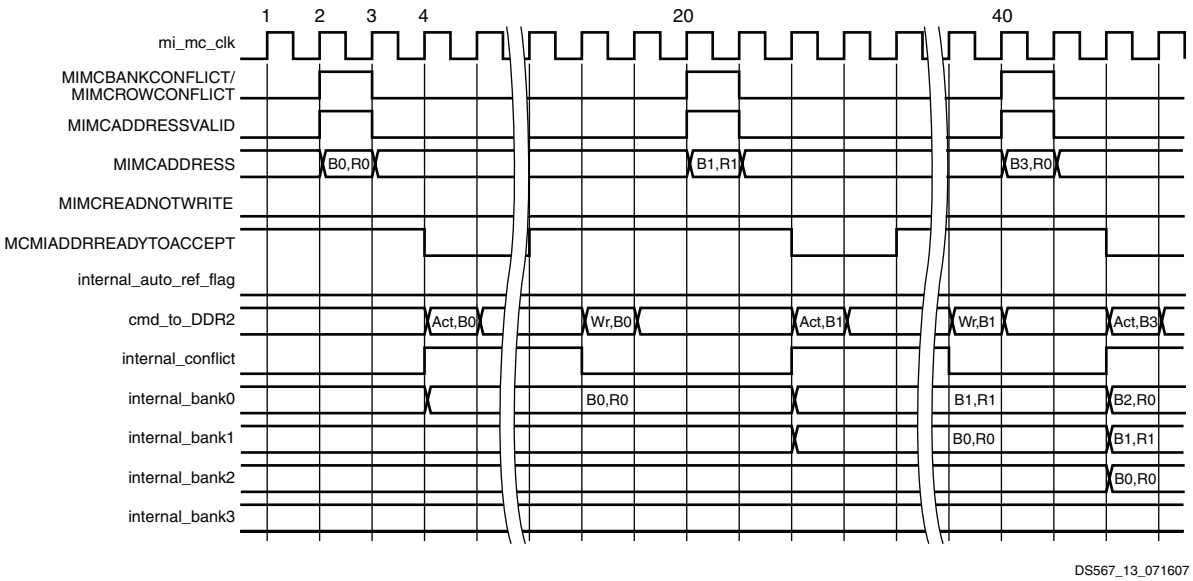


*Figure 12:* **Bank Management with No Internal Banks Open**

In Figure 13, 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.

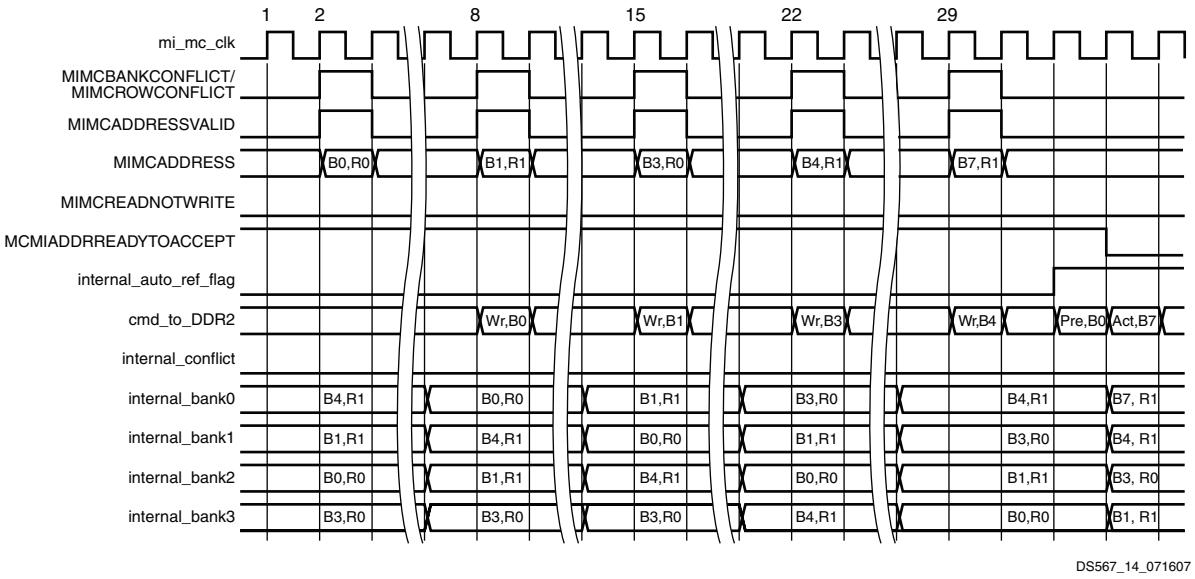


*Figure 13:* **Bank Management with Four Internal Banks Open**

In Figure 14, 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.

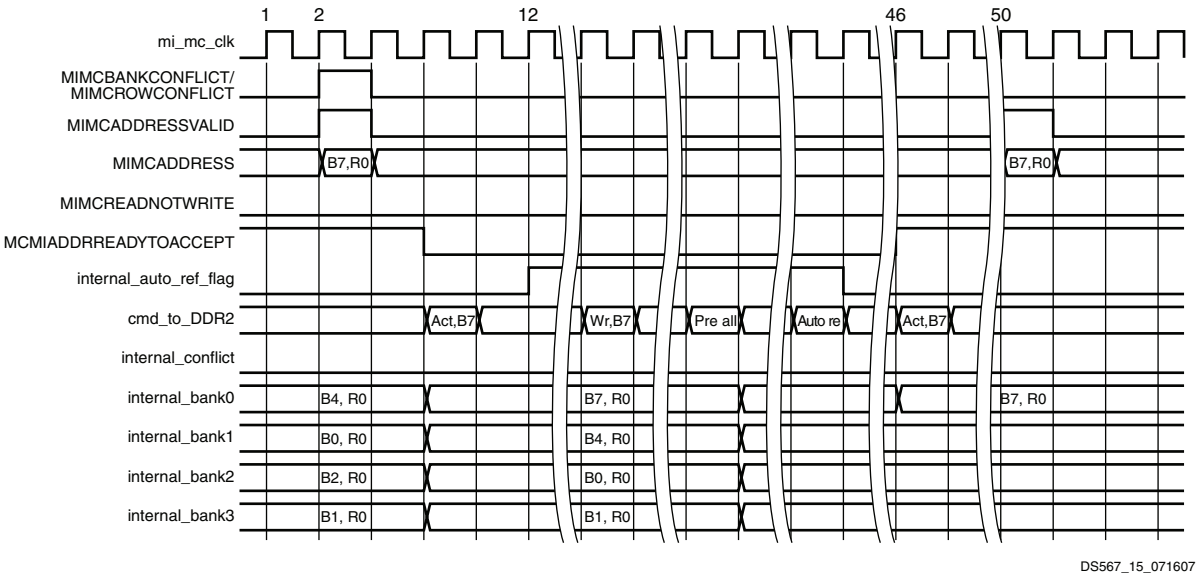


*Figure 14:* **Bank Management During Auto Refresh**

## ECC Feature

The built in Hamming code error correction feature in the Virtex-5 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.

The MCI parameter MIB_RMW_ENABLE must be set to 1'b1 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 in order 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.

### Auto Hold Off and Write Data Delay Values in MCI

The controller is designed to work with an auto hold off value of 2 (four PPC440MC DDR2 Memory Controller clocks) and a write data delay value of 0 (zero clocks). These values have to be set in the MCI for optimal configuration.

## Target Technology

The intended target technology is a Virtex-5 FXT FPGA.

### 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

| Date | Version | Revision |
|---|---|---|
| 01/15/2008 | 1.0 | Initial Xilinx release with ISE™ 10.1 EDK software release. |
| 03/19/2008 | 1.1 | Release with ISE 10.1 SP1 EDK software release. |
| 03/31/2008 | 1.1.1 | Minor publication issue. |
| 04/23/2008 | 1.2 | Revised version of reference design to v1_02_a. Changed frequency for 72-bit C_DDR_DWIDTH in Table 9. |

## Notice of Disclaimer