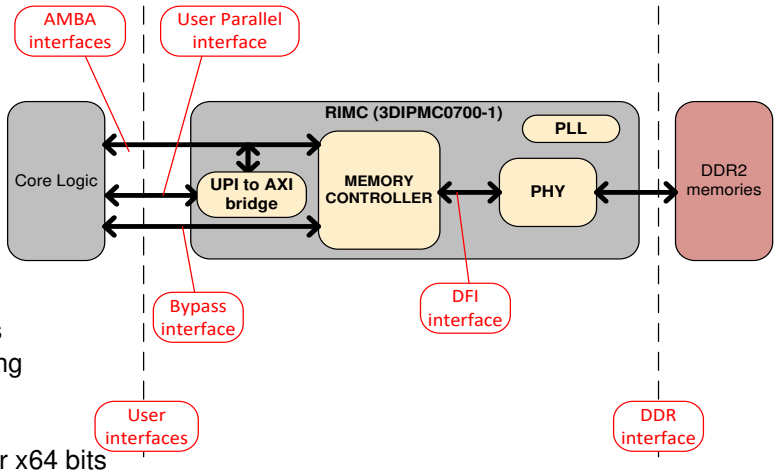
# 1 FEATURES

- Configurable via AMBA compliant (AXI/AHB/APB) user interface
- DFI 2.1 compliant DDR PHY interface
- Provides rank and bank management algorithms
- Dynamically configurable via an 8-bit APB slave interface
- Controller Bypass mode provides direct access to memories
- Selectable Hamming or Reed-Solomon Error Correction Code (ECC)
- Configurable DDR2 ranks to increase memory capacity
- Clock & ODT settings compatible with 3D PLUS modules
- Selectable Burst-of-4 or Burst-of-8 accesses
- Selectable Enable/Disable of DDR2 scrubbing
- User-definable scrubbing frequency
- Selectable DRAM refresh time
- Configurable user data width: x8, x16, x32 or x64 bits




# 2 OVERVIEW

The Radiation Intelligent Memory Controller (RIMC) is a fully configurable DDR2 SDRAM memory controller IP core designed to work with 3D PLUS DDR2 memory modules to achieve improved radiation tolerance. The RIMC contains all of the standard functions of a DDR memory controller for data width applications from 8b up to 64b, and additional functions, including Single Event Upset (SEU) mitigation and Single Event Functional Interrupt (SEFI) protection (later described in paragraph 4.2) to be able to work in radiation environments.

In addition to the memory controller IP within the RIMC IP core, the user can optionally select any of a number of 3D PLUS provided PHY IP or the user can develop their own PHY IP.

The RIMC has two primary interfaces, the user interface (UIF) and the DDR memory interface (DIF).

The UIF is comprised of at least one AHB bus, AXI bus or MIG, and also contains an APB bus for user dynamic configuration:

- Slave APB interface dedicated to internal registers
- Optional slave AHB/AXI interface
- Optional Xilinx MIG interface
- Optional Bypass interface

The Memory Controller is defined by the previously described UIF and the DIF, compliant to DFI 2.1, to send commands and data to the DDR memory components through the DDR PHY (which is uniquely configured to work with specific FPGAs/ASICs).

# Table Of Content

# TABLES and FIGURES

# 3 RIMC CONFIGURATION

To use the RIMC to access 3D PLUS DDR2 Memory modules, it must first be setup by writing data to dedicated RIMC configuration registers. This paragraph describes the flow to configure these parameters in the RIMC and how to build RIMC initialization circuitry.

## Memory Setting

The Memory Setting configuration data are dependent on the 3D PLUS memory modules' characteristics, which can be found within 3D PLUS's memory module Detail Specifications. Below are the items that should be configured.

- The burst length (BURST_LEN generic) shall be set to select either a Burst-of-4 or a Burst-of-8
- The user data bus width (DATA_WIDTH generic) should be selected by the application, and the ECC width (ECC_WIDTH generic) should be selected in accordance with paragraph 4.2.1;
- The number of control signals, ranks, banks and rows shall be configured per Table 30.

The other configuration data are dependent on the end-user's application.

## Clock Frequency

The Clock should be configured in accordance with paragraph 4.1;

## Error Management

Error Management should be configured in accordance with paragraph 4.2; Below are the items which should be configured.

- ECC setting
- Scrubbing setting
- SEFI protection setting

## Interface Setting

Interface Setting should be configured in accordance with paragraph 4.3. Below are the items which should be configured.

- Core logic Interface AXI/AHB/APB
- DFI Interface
- Bypass mode (Direct DDR2 Access)
- Uncorrectable errors Interruption

# 4 MEMORY CONTROLLER

In addition to all of the standard functions of an Synchronous DRAM memory controller, the Memory controller contains the functions described in the following paragraphs.

The RIMC IP is responsible for the following standard functions:

- Bank management
- Conversion of user AMBA commands to DFI compatible DDR commands.

The RIMC IP also embeds the following additional functions:

- Data error management including an ECC and a scrubbing mechanism;
- SEFI detection and correction mechanisms;

## 4.1 CLOCK, RESET AND INITIALIZATION

The RIMC needs 3 internal clocks. The first clock, SysClk, is dedicated to the Memory Controller as well as the AMBA user interfaces. The second clock, DFIClk, is dedicated to the DFI interface. These 2 clocks shall be synchronous with a frequency ratio of 1:1, 2:1 or 4:1 (nCK_PER_CLK). The third clock has a 200MHz fixed frequency, and is used for the Xilinx PHY only (this clock is not requested for RTG4).

DFIClk and SysClk are generated by an RIMC embedded PLL from ClkIn.

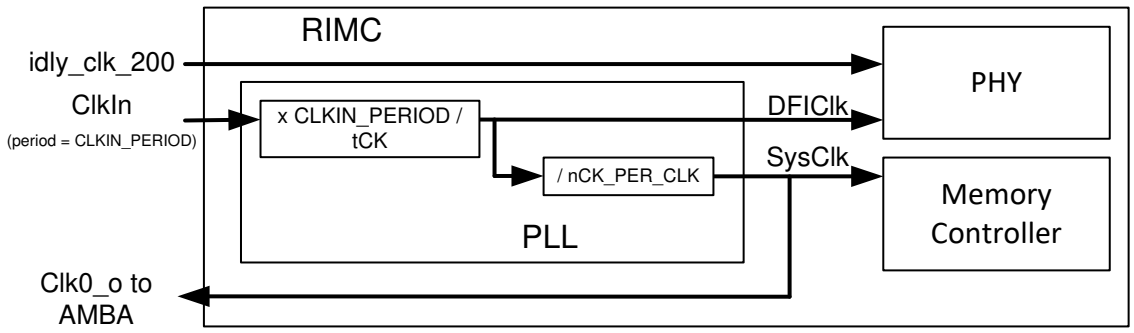



Figure 1: Embedded clock management

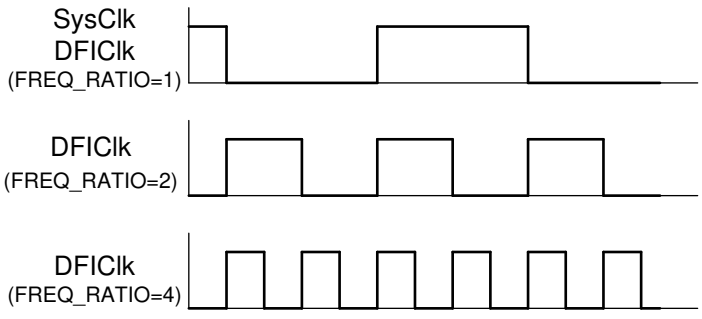



Figure 2: nCK_PER_CLK values

| FPGA Type | nCK_PER_CLK |
|---|---|
| Xilinx V5 | 1, 2, 4 |
| Xilinx 7-series | 2, 4 |
| Microsemi RTG4 | 1, 2, 4 |

Table 1: Frequency ratio generics configuration

One asynchronous input reset (Rst_N) must be provided.

After deassertion of the reset input port (Rst_N), the RIMC automatically configures itself by reading the following configuration input pins:

- SCRUB_EN: This input port can enable or disable the scrubbing mechanism. When activated the scrubbing performs read and write accesses across the entire memory array after the initialization phase has been completed.
- REF_MODE: This input port is used to select 32ms or 64ms refresh time

After this automatic configuration process, the user may dynamically update the internal configuration using the APB slave interface.

A memory initialization sequence is recommended after powering up the User Memory, and this initialization sequence can be enabled through the INIT_Pad APB register. When enabled, the RIMC IP Core initializes the entire memory array with a fixed value as soon as the Physical layer is ready (after the completion of the Power-up sequence and calibration phase). The Byte value to be written to the entire memory array is contained in the Init_Reg register.

If the initialization phase is not performed and a Read is requested by the user to an address prior to a Write at the same address, the RIMC IP Core may detect potential ECC errors.

## 4.2 ERROR MANAGEMENT

### 4.2.1 ERROR CORRECTION CODE (ECC)

The RIMC SEU mitigation scheme uses different configurable ECCs (Hamming or Reed-Solomon) and scrubbing to correct SEUs and Single Event Row Errors (SERE). For example, using a Reed-Solomon code for 32b data and 50% overhead [2 * RS(6;4), m=4, Global Bus = 48bits], the RIMC can correct up to 8 bits of error (row error) in one die per 48b, and 2 SEUs in the same address of different dice per 48b. In cases where scrubbing is applied, the worst case (in which one particle creates 2 upsets in 2 dice) error rate will be 3.8E-9 upset/day/module.

The RIMC implements an Error Correction Code (ECC) with additional DDR2 components. The supported configurations are described in Table 2.

Two different ECC types are handled by the RIMC:

- Hamming, which is able to mitigate SEU
- Reed-Solomon, which is able to mitigate SEU, MBU and SERE.

| DDR2 Component Width | User Data Bus Width (generic DATA_WIDTH) | ECC Bus Width (generic ECC_WIDTH) | Global Bus Width (DQ_WIDTH = DATA_WIDTH+ECC_WIDTH) | ECC onfiguration |
|---|---|---|---|---|
| 8 | 8 | 0 | 8 | No ECC |
| 8 | 8 | 5 | 16 | Hamming 8+5 (1) |
| 8 | 16 | 0 | 16 | No ECC |
| 8 | 16 | 6 | 24 | Hamming 16+6 (1) |
| 8 | 16 | 16 | 32 | 2*RS(6;4) with m=4 |
| 8 | 32 | 0 | 32 | No ECC |
| 8 | 32 | 7 | 40 | Hamming 32+7 (1) |
| 8 | 32 | 16 | 48 | 2*RS(6;4) with m=4 |
| 8 | 64 | 0 | 64 | No ECC |
| 8 | 64 | 32 | 96 | 4*RS(6;4) with m=4 |
| 16 | 16 | 0 | 16 | No ECC |
| 16 | 32 | 0 | 32 | No ECC |
| 16 | 32 | 32 | 64 | 4*RS(4;2) with m=4 |
| 16 | 64 | 8 | 72 | Hamming 64+8 (1) |
| 16 | 64 | 32 | 96 | 4*RS(6;4) with m=4 |

Table 2: Supported ECC codes

*Note 1: This ECC utlilizes a Hamming code. This code CANNOT correct for a DDR2 component failure but it is able to correct a SEU.*

To use this feature additional DDR2 components are added in parallel (data path increased) on the board.

The RIMC can be used with 8-bit and 16-bit DDR2 component widths.

Below is a list of available 3D PLUS DDR2 modules with their configurations:

| DDR II P/N | Density | Configuration | Access/Clock | Package |
|---|---|---|---|---|
| 3D2D1G08US1285 | 1G | 128M x 8 | 200-333Mhz | SOP74 |
| 3D2D2G08US2662 | 2G | 256M x 8 | 200-333Mhz | SOP74 |
| 3D2D2G16UB2684 | 2G | 128M x 16 | 200-333Mhz | BGA95 |
| 3D2D4G08US4661 | 4G | 512M x 8 | 200-333Mhz | SOP74 |
| 3D2D4G72UB3652 | 4G | 64M x 72 | 200-333Mhz | BGA191 |
| 3D2D6G48UB3687 | 6G | 128M x 48 | 200-333Mhz | BGA143 |
| 3D2D6G48UQ3694 | 6G | 128M x 48 | 200-333Mhz | QFP144 |
| 3D2D8G08US8663 | 8G | 1G x 8 | 200-333Mhz | SOP88 |

Table 3: 3D PLUS DDR2 modules

### 4.2.2 SCRUBBING FUNCTIONALITY

A scrubbing mechanism can be dynamically enabled with the SCRUB_EN input pin or the ScrubEn_Reg register.

When enabled, the RIMC IP core periodically reads the entire memory array and then writes the corrected data back into the memory array. The frequency of the scrubbing is user-defined through the Scrub_Reg register.

An Example of Scrub_Reg configuration:

- The RIMC is connected to a single 3D2D6G48UB3687 (BGA package) or a single 3D2D6G48UQ3694 (QFP package) organized as 128M x 48bits
- User wants to scrub the full memory array in 60 seconds (t).
- DFIClk frequency ($F_{DFI}$) is 266 MHz
- Burst length is set to 8

Scrub_Reg = BURST_LENGTH * t * $F_{DFI}$ / 128M = 953 clock cycles

In this example the RIMC performs one Read/Write Burst every 953 clock cycles.

### 4.2.3 SEFI PROTECTION

Traditionally, SEFI mitigation has involved power cycling after the occurrence of a SEFI; however, power cycling will lead to data loss. To avoid data loss, a specific SEFI protection technique has been designed in the RIMC IP Core to prevent SEFI and to replace the traditional power cycling strategy. This SEFI protection is integrated in the RIMC IP core and is transparent to the user. Verification tests have been performed to confirm the robustness of this protection IP, and no SEFI were observed up to LET>60Mev-cm2/mg.

To configurate the device SEFI protection, the Bypass_Reg register should be configured as below:

b00: SEFI protection set to Strong Level
b01: SEFI protection set to Simple Level
b10: Switch off the SEFI protection

## 4.3 RIMC INTERFACES

The RIMC is defined by three interfaces (see Figure 3):

- The AMBA interface
- The Direct DDR Access interface
- The DDR PHY interface, compliant to DFI 2.1

These three interfaces are described in the following paragraphs.

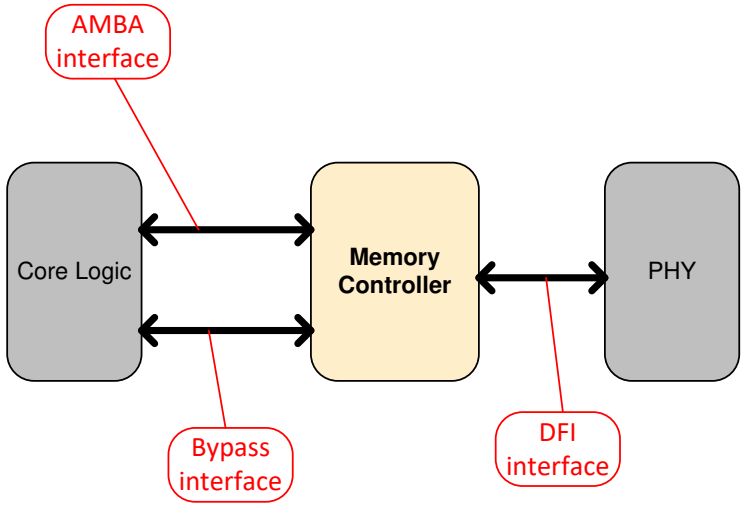



Figure 3: Interfaces with DDR controller

### 4.3.1 AMBA INTERFACES

Three AMBA interfaces are available for connection to the Memory Controller:

- AMBA AXI: 0 to 8 ports can be instantiated through the NB_AXI_SLV generic.
- AMBA AHB: 0 to 8 ports can be instantiated through the NB_AHB_SLV generic
- AMBA APB: This interface is used to configure the IP.

At least one port shall be instantiated to be synthesizable, i.e. NB_AXI_SLV + NB_AHB_SLV >= 1

The Memory Controller can be configured by the core logic using the slave APB interface dedicated to internal registers.

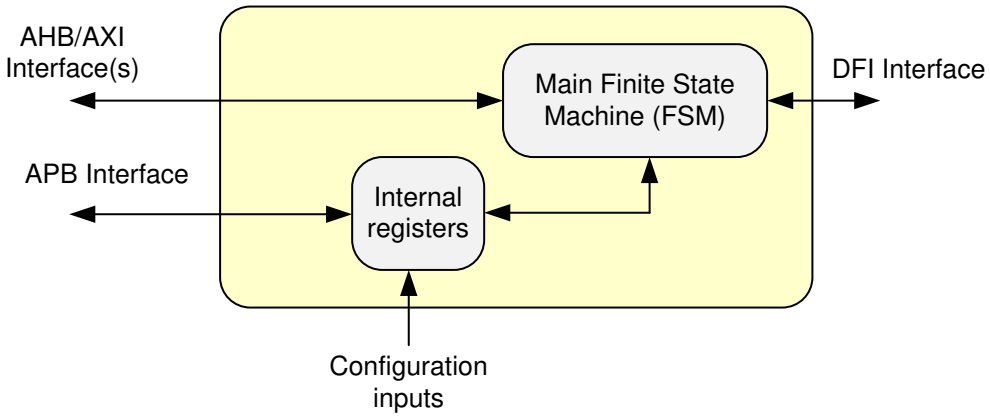



Figure 4: Configuration management

#### *4.3.1.1 AXI interface*

The Memory Controller is compliant to the AMBA Advanced eXtensible Interface (AXI) protocol. Up to 8 slaves AXI interfaces can be instantiated inside the RIMC through the NB_AXI_SLV generic.

The data bus width of both the write channel and the read channel can be configured to 16, 32, 64, 128 or 256 bits.

*Note: To realize optimal bandwidth, AXI data width should be greater than or equal to "DDR data bus" width * 2 * nCK_PER_CLK.*

The key features of the AXI protocol are:

- separate address/control and data phases
- support for unaligned data transfers using byte strobes
- burst-based transactions with only start address issued
- separate read and write data channels to enable Direct Memory Access (DMA)
- ability to issue multiple outstanding addresses
- out-of-order transaction completion
- easy addition of register stages to provide timing closure.

#### *4.3.1.2 AHB interface*

The Memory Controller is compliant with AHB slave interface data widths of 16, 32, 64, 128 and 256 bits.

*Note: To realize optimal bandwidth, AHB data width should be greater than or equal to "DDR data bus" width * 2 * nCK_PER_CLK.*

The AHB slave interface provides the following input signals for each AHB port:

- HSEL: Slave select
- HADDR: Address bus
- HWRITE: Transfer direction (0 for Read / 1 for Write)
- HTRANS: Transfer type
- HSIZE: Transfer size
- HBURST: Burst type
- HWDATA: Write data bus
- HPROT: Protection control. This signal is not used by RIMC IP Core
- HREADY: Transfer done
- HMASTER: Master number. This signal is not used by RIMC IP Core
- HMASTLOCK: Locked sequence

The AHB slave interface provides the following output signals for each AHB port:

- HREADY: Transfer done
- HRESP: Transfer response
- HRDATA: Read data bus
- HSPLIT: Split completion request. The RIMC does not implement the SPLIT functionality so this output is driven LOW.

The slave AHB accepts the following burst types:

- SINGLE
- INCR4
- INCR8
- INCR16

The Memory Controller implements neither the RETRY nor the SPLIT functions. The Memory Controller does not implement the BURST with an unknown number of beats.

An HRESP_ERROR is generated by the RIMC in the following cases:

- The DDR2 memory array is not initialized
- An uncorrectable error is detected during a Read operation.
- An AHB address is out of the DDR2 address range

#### *4.3.1.3 APB interface*

An APB bus is instantiated inside the Memory Controller.

The Memory Controller is compliant with the slave APB interface data width of 8 bits.

The APB interface can be used to configure the Memory Controller internal registers (the complete list of registers is provided in paragraph 4.5).

### 4.3.2 BYPASS INTERFACE

In addition to the AHB/AXI interfaces the Memory Controller provides a direct path to the DDR2 memory array in a *Controller Bypass* mode.

The following signals are provided at the user interface:

- user_ras_n
- user_cas_n
- user_we_n
- user_addr
- user_ba
- user_cs_n
- user_cke
- user_odt
- user_reset_n
- user_rddata_en
- user_dqiv
- user_dqi
- user_wdata_en
- user_dqo
- user_dm

When using this interface, signals are transferred directly to the DDR2 memory array without any reformatting inside the Memory Controller, therefore it is the user's responsibility to insure that the signals are in conformance with the applicable DDR2 memory datasheets/Detail Specification.

### 4.3.3 DFI INTERFACE

The Memory Controller implements a DDR PHY Interface (DFI) compatible to DFI 2.1. The DFI is a standardized interface that defines the connectivity between a DDR memory controller and a DDR physical interface (PHY) for DDR1, DDR2, LPDDR2 and DDR3.

PHY IP is available from 3D PLUS, supporting multiple targets (see paragraph 8)

## 4.4 MEMORY CONTROLLER GENERICS/PARAMETERS AND PORTS

The Memory Controller contains the ports defined in Table 4 to Table 7.

The Memory Controller contains the generics/parameters defined in Table 8.

| Name | Direction | Comments |
|---|---|---|
| SysClk | In | Main input clock used for all of the RIMC (except the DFI interface) |
| DFIClk | In | Input clock used for DFI interface. Can be SysClk, 2* SysClk or 4* SysClk frequency |
| Rst_N | In | Input Reset - synchronous to SysClk, active LOW |
| DFIRst_N | In | Input Reset - synchronous to DFIClk, active LOW |
| SCRUB_EN | In | 0: Scrubbing is disabled<br>1: Scrubbing is enabled |
| REF_MODE | In | 0: Normal refresh<br>1: Refresh timing is divided by 2 |

Table 4: Memory Controller Core Ports

| Name | Direction | Comments |
|---|---|---|
| Ahbsi(NB_AHB_SLV-1:0) | In | AHB slave input array of record type |
| Ahbso(NB_AHB_SLV-1:0) | Out | AHB slave output array of record type |
| AXI_aw_out(NB_AXI_SLV-1:0) | In | AXI write address channel input array of record type |
| AXI_aw_in(NB_AXI_SLV-1:0) | Out | AXI write address channel output array of record type |
| AXI_w_out(NB_AXI_SLV-1:0) | In | AXI write data channel input array of record type |
| AXI_w_in(NB_AXI_SLV-1:0) | Out | AXI write data channel output array of record type |
| AXI_b_out(NB_AXI_SLV-1:0) | Out | AXI write response channel input array of record type |
| AXI_b_in(NB_AXI_SLV-1:0) | In | AXI write response channel output array of record type |
| AXI_ar_out(NB_AXI_SLV-1:0) | In | AXI read address channel input array of record type |
| AXI_ar_in(NB_AXI_SLV-1:0) | Out | AXI read address channel output array of record type |
| AXI_r_out(NB_AXI_SLV-1:0) | Out | AXI read data channel input array of record type |
| AXI_r_in(NB_AXI_SLV-1:0) | In | AXI read data channel output array of record type |
| Apbi | In | APB slave input |
| Apbo | Out | APB slave output |

Table 5: Memory Controller AMBA Ports

| Name | Direction | Comments |
|---|---|---|
| **Control Interface** | | |
| Dfi_address | Out | DFI address bus |
| Dfi_bank | Out | DFI bank bus |
| Dfi_cas_n | Out | DFI column address strobe bus |
| Dfi_cke | Out | DFI clock enable bus |
| Dfi_cs_n | Out | DFI chip select bus |
| Dfi_odt | Out | DFI on-die-termination control bus |
| Dfi_ras_n | Out | DFI row address strobe bus |
| Dfi_reset_n | Out | DFI reset bus |
| Dfi_we_n | Out | DFI write enable bus |
| **Write data interface** | | |
| Dfi_wrdata | Out | DFI write data bus |
| Dfi_wrdata_en | Out | DFI write data and data mask enable |
| Dfi_wrdata_mask | Out | DFI write data Byte mask |
| **Read data interface** | | |
| Dfi_rddata_en | Out | DFI read data enable |
| Dfi_rddata | In | DFI read data bus |
| Dfi_rddata_dnv | In | DFI data not valid |

| Name | Direction | Comments |
|---|---|---|
| Dfi_rddata_valid | In | DFI read data valid indicator |
| **Update interface** | | |
| Dfi_ctrlupd_req | Out | MC-initiated update request |
| Dfi_phyupd_ack | Out | PHY-initiated update acknowledge |
| Dfi_ctrlupd_ack | In | MC-initiated update acknowledge |
| Dfi_phyupd_req | In | PHY-initiated update request |
| Dfi_phyupd_type | In | PHY-initiated update select |
| **Status interface** | | |
| Dfi_dram_clk_disable | Out | DRAM clock disable |
| Dfi_freq_ratio | Out | DFI frequency ratio indicator |
| Dfi_init_complete | In | PHY initialization complete |
| **Training interface** | | |
| Dfi_rdlvl_load | Out | Not used for DDR2. |
| Dfi_rdlvl_cs_n | Out | Not used for DDR2. |
| Dfi_rdlvl_en | Out | Not used for DDR2. |
| Dfi_rdlvl_edge | Out | Not used for DDR2. |
| Dfi_rdlvl_delay_X | Out | Not used for DDR2. |
| dfi_rdlvl_gate_en | Out | Not used for DDR2. |
| dfi_rdlvl_gate_delay_X | Out | Not used for DDR2. |
| dfi_wrlvl_load | Out | Not used for DDR2. |
| dfi_wrlvl_cs_n | Out | Not used for DDR2. |
| dfi_wrlvl_strobe | Out | Not used for DDR2. |
| dfi_wrlvl_en | Out | Not used for DDR2. |
| dfi_wrlvl_delay_X | Out | Not used for DDR2. |
| dfi_rdlvl_resp | In | Not used for DDR2. |
| dfi_rdlvl_mode | In | Not used for DDR2. |
| dfi_rdlvl_gate_mode | In | Not used for DDR2. |
| dfi_wrlvl_mode | In | Not used for DDR2. |
| dfi_wrlvl_resp | In | Not used for DDR2. |

Table 6: Memory Controller DFI Ports

| Name | Direction | Comments |
|---|---|---|
| user_ras_n | In | Row Address Select. Used only in the “Controller Bypass” mode |
| user_cas_n | In | Column Address Select. Used only in the “Controller Bypass” mode |

| Name | Direction | Comments |
|---|---|---|
| user_we_n | In | Write Enable, active LOW. Used only in the “Controller Bypass” mode |
| user_address | In | Address bus. Used only in the “Controller Bypass” mode |
| user_bank | In | Bank Address bus. Used only in the “Controller Bypass” mode |
| user_cs_n | In | Chip Select, active LOW. Used only in the “Controller Bypass” mode |
| uer_cke | In | Clock Enable. Used only in the “Controller Bypass” mode |
| user_odt | In | On Die Termination. Used only in the “Controller Bypass” mode |
| user_reset_n | In | Not used for DDR2. |
| user_rddata_en | In | Read data enable |
| user_dqiv | Out | Data input valid. Used only in the “Controller Bypass” mode |
| user_dqi | Out | Data input. Used only in the “Controller Bypass” mode |
| user_wdata_en | In | Write data enable |
| user_dqo | In | Data output. Used only in the “Controller Bypass” mode |
| user_dm | In | Data output mask. Used only in the “Controller Bypass” mode |

Table 7: Memory Controller Bypass Ports

| Name | Comments | Default value |
|---|---|---|
| DRAM_TYPE | Shall be set to “DDR2” | “DDR2” |
| RTT_NOM | Nominal ODT termination value | 75 |
| AL | Additive Latency option | 0 |
| OUTPUT_DRV | DDR2/DDR3 output drive. | “HIGH” |
| HDMAX | Width of the user AMBA busses | 32 |
| GSYNCRST | 0: Asynchronsous reset<br>1: Synchronous reset | 1 |
| G_RAM_INST | 0: RAM is inferred<br>1: RAM is instantiated. In this configuration, the VHDL file ip/tpram_inst.vhd shall be updated by the customer | 0 |
| USE_DFF | 0: Use block ram inside RIMC<br>1: Use DFF instead of block RAM<br>2: Automatic selection. DFF will be used for memories less than 600 bits | 0 |

| Name | Comments | Default value |
|---|---|---|
| ECC_EN | 0: No ECC for internal RAMs<br>1: Hamming ECC for internal RAMs | 0 |
| ENDIANNESS | Endianness of the AHB bus. Can be set to BIG_ENDIAN or LITTLE_ENDIAN. This generic is not meaningful for AXI bus. | BIG_ENDIAN |
| SIMU | Set to 1 for simulation only | 0 |
| NB_AHB_SLV | Number of slave AHB busses. Shall be between 0 and 8. | 1 |
| NB_AXI_SLV | Number of slave AXI busses. Shall be between 0 and 8. | 1 |
| tphy_wrdata | | 0 |
| tphy_wrlat | Number of DFI clock cycles between a write command and the wrdata_en | 0 |
| trddata_en | | 0 |
| TWR_PS | Write recovery time, in ps | 15000 |
| TWTR_PS | Write to read timing, in ps | 7500 |
| TRTP_PS | Internal READ to PRECHARGE delay, in ps | 7500 |
| TRFC_PS | REFRESH to ACTIVE command interval, in ps | 127500 |
| CLK_PERIOD | SysClk clock period, in ps | 3000 |
| nCK_PER_CLK | Ratio between SysClk and DDR2 memory clock (see Table 1 for allowed values) | 1 |
| CAS_LAT | Read CAS latency. In number of clock cycles | 5 |
| TWO_T_TIME_EN | Control to DDR2 can be performed in “2T” mode<br>0: Mode “1T”<br>1: Mode “2T” | 0 |
| SLOT_NUM | This generic defines the slot/phase where the DFI command is applied in the DFI bus. It should be set to 0 for all targets except Xilinx 7-series FPGA.<br>It shall be in the range from 0 to nCK_PER_CLK-1 | 0 |
| BURST_TYPE | 0=sequential, 1=interleaved | 0 |
| BURST_LEN | Shall be 4 or 8 | 8 |
| CS_NUM | Number of ranks. Shall be set to 1, 2, 4 or 8 | 1 |
| BANK_WIDTH | Number of banks | 3 |
| ROW_WIDTH | Number of rows | 14 |

| Name | Comments | Default value |
|---|---|---|
| COL_WIDTH | Number of columns | 10 |
| DATA_WIDTH | Data width, without ECC. See Table 1 for authorized values. | 16 |
| ECC_WIDTH | Width of the ECC | 6 |
| DQ_WIDTH | Width of the global DQ bus (DQ_WIDTH shall be DATA_WIDTH+ECC_WIDTH, rounded up to the enxt multiple of 8) | 24 |

Table 8: Memory Controller Generic/Parameter list

### 4.5 MEMORY CONTROLLER REGISTERS

The Memory Controller contains the registers defined in Table 9. Details are given from Table 10 to Table 24.

| Registers | Address |
|---|---|
| ECC_DErr_Reg0 | 0x00 |
| ECC_DErr_Reg1 | 0x01 |
| ECC_SErr_Reg0 | 0x02 |
| ECC_SErr_Reg1 | 0x03 |
| ECC_DErr_Clr | 0x04 |
| ECC_SErr_Clr | 0x05 |
| Bypass_Reg | 0x06 |
| Init_Pad_Reg | 0x07 |
| Init_Pad_Sts | 0x08 |
| Init_Pad_Ctl | 0x09 |
| Scrubbing_Reg0 | 0x0A |
| Scrubbing_Reg1 | 0x0B |
| ScrubEn_Reg | 0x0C |
| Scrub_Ctl | 0x0D |
| CorEn_Reg | 0x0E |
| CorBeat_Reg | 0x0F |
| CorVec_Reg0 | 0x10 |
| CorVec_Reg1 | 0x11 |
| CorVec_Reg2 | 0x12 |
| CorVec_Reg3 | 0x13 |
| CorVec_Reg4 | 0x14 |
| CorVec_Reg5 | 0x15 |
| CorVec_Reg6 | 0x16 |
| CorVec_Reg7 | 0x17 |
| CorVec_Reg8 | 0x18 |
| CorVec_Reg9 | 0x19 |
| CorVec_Reg10 | 0x1A |
| CorVec_Reg11 | 0x1B |
| ScrubAd_Sts0 | 0x1C |
| ScrubAd_Sts1 | 0x1D |
| ScrubAd_Sts2 | 0x1E |
| ScrubAd_Sts3 | 0x1F |
| Version_Sts | 0x20 |

Table 9: RIMC registers list

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 15-0 | CNT | Uncorrectable ECC counter | R | 0 |

Table 10: ECC_DErr_Reg0 (Most Significant Byte) and ECC_DErr_Reg1 (Least Significant Byte)

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 15-0 | CNT | Correctable ECC counter | R | 0 |

Table 11: ECC_SErr_Reg0 and ECC_SErr_Reg1

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | CLR | 0: No effect<br>1: Clear ECC_DErr_Reg registers | W | 0 |

Table 12: ECC_DErr_Clr register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | CLR | 0: No effect<br>1: Clear ECC_SErr_Reg registers | W | 0 |

Table 13: ECC_SErr_Clr register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-4 | - | Reserved | R | 0 |
| 3 | - | Reserved. Do NOT modify | R/W | 1 |
| 2-1 | MODE | b00: SEFI protection = STRONG<br>b01: SEFI protection = SIMPLE<br>b10: No SEFI protection<br>b11: Reserved | R/W | 0 |
| 0 | BYP | 0: Normal mode<br>1: Bypass mode. In this mode the only available user interface is the bypass interface | R/W | 0 |

Table 14: Bypass_Reg register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-0 | VAL | Byte value used for initialization of the entire memory array. | R/W | 0 |

Table 15: Init_Pad_Reg register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | STS | 0: Padding is in IDLE<br>1: Padding is in progress | R | 0 |

Table 16: Init_Pad_Sts register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | START | 0: No effect<br>1: Begin the padding of the entire memory array | W | 0 |

Table 17: Init_Pad_Ctl register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | EN | 0: Scrubbing disabled<br>1: Scrubbing enabled | R/W | 0 |

Table 18: ScrubEn_Reg register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 15-0 | FREQ | Register used to select the frequency of the scrubbing.<br>1 DDR Burst performed each FREQ SysClk clock cycles.<br>Note: This value shall be greater than or equal to 256 | R/W | 0x0100 |

Table 19: Scrubbing_Reg0 and Scrubbing_Reg1

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | RST | 0: No effect<br>1: Reset ScrubAd_Sts0 to ScrubAd_Sts3 registers | W | 0 |

Table 20: Scrub_Ctl register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 31-0 | AD | Address of the memory in which the scrubbing detected the 1st error | R | 0 |

Table 21: ScrubAd_Sts0 to ScrubAd_Sts3

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-1 | - | Reserved | R | 0 |
| 0 | EN | Enable the corruption of data. Used to test the correct behavior of the ECC mechanism | R/W | 0 |

Table 22: CorEn_Reg register

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 95-0 | COR_BIT | Bit to corrupt (used only when CorEn_Reg register = 0x01) | R/W | 0 |

Table 23: CorVec_Reg0 to CorVec_Reg11

| Bits | Field Name | Comments | R/W | Reset value |
|---|---|---|---|---|
| 7-3 (1) | - | Reserved | R | 0 |
| 2-0 (1) | COR_BEAT | Beat to corrupt (used only when CorEn_Reg register = 0x01) | R/W | 0 |

Table 24: CorBeat_Reg

Note 1: The width of COR_BEAT field is log2(2 x nCK_PER_CLK), which means:
- COR_BEAT(0:0) when nCK_PER_CLK=1
- COR_BEAT(1:0) when nCK_PER_CLK=2
- COR_BEAT(2:0) when nCK_PER_CLK=4

# 5 RIMC (MEMORY CONTROLLER + PHY)

The Physical layer IP (PHY) is dependant of the FPGA/ASIC technology being used. The following PHY are available for use inside the RIMC:

- Xilinx Virtex-5 FPGA
- Xilinx 7-series FPGA
- Microsemi RTG4 FPGA

## 5.1 RIMC INTERFACES

The RIMC provides the following interfaces:
- The AMBA interfaces (already described in paragraph 4.3.1)
- The Bypass interface (already described in paragraph 4.3.2)
- The DDR2 memory interface
- The Xilinx MIG interface

### 5.1.1 DDR2 INTERFACE

The RIMC is compliant to the JEDEC DDR2 memory interface specification.

### 5.1.2 XILINX MIG INTERFACE

A specific MIG interface is present at the UI of the RIMC to allow a direct replacement of a MIG DDR controller from Xilinx with the RIMC.
A bridge is provided inside of the RIMC to convert the MIG interface into a standard AXI interface.

## 5.2 RIMC GENERICS/PARAMETERS AND PORTS

The RIMC contains the ports defined in Table 25 to Table 29.

The RIMC contains the generics defined in Table 30.

| Name | Direction | Comments |
|---|---|---|
| ClkIn | In | Input clock. The “Memory controller” and PHY input clocks are derivated from this clock to operate with the nCK_PER_CLK generic. |
| idly_clk_200 | In | Input clock at 200 MHz. This clock is used only for Xilinx FPGAs. |
| Rst_N | In | Input reset, active LOW |
| clk0_o | Out | User output clock. This clock shall be used for User Interfaces (AHB, AXI, APB, Bypass) |
| rst0_n_o | Out | Output reset, synchronous to clk0_o. Active LOW |
| SCRUB_EN | In | 0: Scrubbing is disabled<br>1: Scrubbing is enabled |
| REF_MODE | In | 0: Normal refresh<br>1: Refresh timing is divided by 2 |

Table 25: RIMC Core Ports

| Name | Direction | Comments |
|---|---|---|
| Ahbsi(NB_AHB_SLV-1:0) | In | AHB slave input array of record type |
| Ahbso(NB_AHB_SLV-1:0) | Out | AHB slave output array of record type |
| AXI_aw_out(NB_AXI_SLV-1:0) | In | AXI write address channel input array of record type |
| AXI_aw_in(NB_AXI_SLV-1:0) | Out | AXI write address channel output array of record type |
| AXI_w_out(NB_AXI_SLV-1:0) | In | AXI write data channel input array of record type |
| AXI_w_in(NB_AXI_SLV-1:0) | Out | AXI write data channel output array of record type |
| AXI_b_out(NB_AXI_SLV-1:0) | Out | AXI write response channel input array of record type |
| AXI_b_in(NB_AXI_SLV-1:0) | In | AXI write response channel output array of record type |
| AXI_ar_out(NB_AXI_SLV-1:0) | In | AXI read address channel input array of record type |
| AXI_ar_in(NB_AXI_SLV-1:0) | Out | AXI read address channel output array of record type |
| AXI_r_out(NB_AXI_SLV-1:0) | Out | AXI read data channel input array of record type |
| AXI_r_in(NB_AXI_SLV-1:0) | In | AXI read data channel output array of record type |
| Apbi | In | APB slave input |
| Apbo | Out | APB slave output |

Table 26: RIMC AMBA Ports

| Name | Direction | Comments |
|---|---|---|
| ddr_ck | Out | DDR clock |
| ddr_ck_n | Out | DDR negative clock |
| ddr_addr | Out | DDR address bus |
| ddr_ba | Out | DDR bank bus |
| ddr_cas_n | Out | DDR column address strobe bus |
| ddr_cke | Out | DDR clock enable bus |
| ddr_cs_n | Out | DDR chip select bus |
| ddr_odt | Out | DDR on-die-termination control bus |
| ddr_ras_n | Out | DDR row address strobe bus |
| ddr_rst_n | Out | DDR reset bus. Not used for DDR2 |
| ddr_dm | Out | DDR data mask |
| ddr_dq | Inout | DDR data |
| ddr_dqs | Inout | DDR data strobe |
| ddr_dqs_n | Inout | DDR negative data strobe |
| ddr_we_n | Out | DDR write enable |

Table 27: RIMC DDR Ports

| Name | Direction | Comments |
|---|---|---|
| app_af_cmd | In | b000 for Write command<br>b001 for Read command<br>Other combinations are invalid |
| app_af_addr | In | Gives information about the address of the memory location to be accessed. This bus contains the bank address, the row address, and the column address |
| app_af_wren | In | Write enable to the User Address FIFO. This signal should be synchronized with the app_af_addr and app_af_cmd signals. |
| app_af_afull | In | Almost Full status of the Address FIFO. When this signal is asserted, the user can write 12 more locations into the FIFO. |
| app_wdf_data | In | User input data. It should contain the fall data and the rise data.<br>Rise data = app_wdf_data[DQ_WIDTH–1: 0]<br>Fall data = app_wdf_data[2*DQ_WIDTH–1: DQ_WIDTH] |
| app_wdf_mask_data | In | User mask data. It should contain the data mask information for both rise and fall data.<br>Rise mask data = app_wdf_mask_data[DM_WIDTH–1: 0]<br>Fall mask data = app_wdf_mask_data[2*DM_WIDTH–1: DM_WIDTH] |
| app_wdf_wren | In | Write enable for the User Write FIFO. This signal should be synchronized with the app_wdf_data and app_wdf_mask_data signals. |
| app_wdf_afull | Out | Almost Full status of the User Write FIFO. When this signal is asserted, the user can write 12 more locations into the FIFO. |
| rd_data_valid | Out | Status signal indicating read data is valid on the read data bus. |
| rd_data_fifo_out | Out | Read data from the memory. |

Table 28: RIMC MIG ports

| Name | Direction | Comments |
|---|---|---|
| user_ras_n | In | Row Address Select. Used only in the “Controller Bypass” mode |
| user_cas_n | In | Column Address Select. Used only in the “Controller Bypass” mode |
| user_we_n | In | Write Enable, active LOW. Used only in the “Controller Bypass” mode |
| user_address | In | Address bus. Used only in the “Controller Bypass” mode |
| user_bank | In | Bank Address bus. Used only in the “Controller Bypass” mode |

| Name | Direction | Comments |
|---|---|---|
| user_cs_n | In | Chip Select, active LOW. Used only in the “Controller Bypass” mode |
| uer_cke | In | Clock Enable. Used only in the “Controller Bypass” mode |
| user_odt | In | On Die Termination. Used only in the “Controller Bypass” mode |
| user_reset_n | In | Not used for DDR2. |
| user_rddata_en | In | Read data enable |
| user_dqiv | Out | Data input valid. Used only in the “Controller Bypass” mode |
| user_dqi | Out | Data input. Used only in the “Controller Bypass” mode |
| user_wdata_en | In | Write data enable |
| user_dqo | In | Data output. Used only in the “Controller Bypass” mode |
| user_dm | In | Data output mask. Used only in the “Controller Bypass” mode |

Table 29: RIMC Bypass Ports

| Name | Comments | Default value |
|---|---|---|
| BA_REPLICATION | Replication of bus ddr_ba. For example, if set to 2, ddr_ba(BANK_WIDTH-1:0) and ddr_ba(2*BANK_WIDTH-1:BANK_WIDTH) outputs the same value | 1 |
| ADDR_REPLICATION | Replication of bus ddr_addr | 1 |
| CMD_REPLICATION | Replication of signals ddr_ras_n, ddr_cas_n, ddr_we_n and ddr_rst_n | 1 |
| SYSCLK_TYPE | Define if a clock buffer shall be added to the SysClk input clock<br>Shall be set to “SINGLE_ENDED” or “NO_BUFFER” | “SINGLE_ENDED” |
| REFCLK_TYPE | Define if a clock buffer shall be added to the idly_clk_200 input clock<br>Shall be set to “SINGLE_ENDED” or “NO_BUFFER” | “SINGLE_ENDED” |
| BANK_TYPE | Used only for Xilinx 7-series PHY.<br>Shall be set to “HP_IO”, “HPL_IO”, “HR_IO” or “HRL_IO” | “HP_IO” |
| DATA_IO_PRIM_TYPE | Used only for Xilinx 7-series PHY.<br>Shall be set to “HP_LP”, “HR_LP” or “DEFAULT” | “HP_LP” |
| DRAM_TYPE | Shall be set to “DDR2” | “DDR2” |
| HDMAX | Width of the user AMBA busses | 32 |
| GSYNCRST | 0: Asynchronsous reset<br>1: Synchronous reset | 1 |

| Name | Comments | Default value |
|---|---|---|
| G_RAM_INST | 0: RAM is inferred<br>1: RAM is instantiated. In this configuration, the VHDL file ip/tpram_inst.vhd shall be updated by the customer | 0 |
| USE_DFF | 0: Use block ram inside RIMC<br>1: Use DFF instead of block RAM<br>2: Automatic selection. DFF will be used for memories less than 600 bits | 0 |
| ECC_EN | 0: No ECC for internal RAMs<br>1: Hamming ECC for internal RAMs | 0 |
| FPGA_SPEED_GRADE | Speed Grade of the FPGA (Used only for Xilinx Virtex-5 FPGA) | 2 |
| ENDIANNESS | Endianness of the AHB bus. Can be set to BIG_ENDIAN or LITTLE_ENDIAN. This generic is not meaningful for AXI bus. | BIG_ENDIAN |
| NB_AHB_SLV | Number of slave AHB busses. Shall be between 0 and 8. | 1 |
| NB_AXI_SLV | Number of slave AXI busses. Shall be between 0 and 8. | 1 |
| SIMU | Set to 1 for simulation only | 0 |
| BURST_LEN | Shall be 4 or 8 | 8 |
| CLKIN_PERIOD | Input clock period, in ps | 3000 |
| tCK | Memory clock period, in ps | 3000 |
| nCK_PER_CLK | Ratio between SysClk and DDR2 memory clock (see Table 1 for allowed values) | 1 |
| CAS_LAT | Read CAS latency. In number of clock cycles | 5 |
| TWO_T_TIME_EN | Control to DDR2 can be performed in “2T” mode<br>0: Mode “1T”<br>1: Mode “2T” | 0 |
| TWR_PS | Write recovery time, in ps | 15000 |
| TWTR_PS | Write to read timing, in ps | 7500 |
| TRTP_PS | Internal READ to PRECHARGE delay, in ps | 7500 |
| TRFC_PS | REFRESH to ACTIVE command interval, in ps | 127500 |
| RTT_NOM | Nominal ODT termination value | “75” |
| RTT_WR | Shall be set to “OFF”, “60” or “120” | “120” |
| OUTPUT_DRV | DDR2/DDR3 output drive. | “HIGH” |
| AL | Additive Latency option | “0” |

| Name | Comments | Default value |
|---|---|---|
| phy_config | Used only for Xilinx 7-series PHY | - |
| CS_NUM | Number of ranks. Shall be set to 1, 2, 4 or 8 | 1 |
| CS_WIDTH | Number od CS_N bits | 1 |
| CKE_WIDTH | Number of CKE bits | 1 |
| ODT_WIDTH | Number of ODT bits | 1 |
| BANK_WIDTH | Number of banks | 3 |
| ROW_WIDTH | Number of rows | 14 |
| COL_WIDTH | Number of columns | 10 |
| CLK_WIDTH | | |
| DATA_WIDTH | Data width, without ECC. See Table 1 for authorized values. | 16 |
| ECC_WIDTH | Width of the ECC | 6 |
| DQ_WIDTH | Width of the DQ bus | 24 |
| DM_WIDTH | Data mask width | 3 |
| DQS_WIDTH | Data strobe width | 3 |

Table 30: RIMC Generic list

# 6 RIMC PERFORMANCES

## 6.1 MAXIMUM FREQUENCY

| | Memory Controller | Physical Layer |
|---|---|---|
| Xilinx Virtex 5 | 100 MHz (Speed Grade -1) (1)<br>125 MHz (Speed Grade -2 and -3) (1) | 333 MHz |
| Xilinx Kintex 7 | 333 MHz | 333 MHz |
| Xilinx Virtex 5 QV (Space Grade) | 83 MHz (1) | 166 MHz |
| Microsemi RTG4 | 62.5 MHz (1) | 200 MHz |

Note 1: For these targets, the memory controller cannot run at the same frequency. The generic nCK_PER_CLK shall be chosen accordingly.

Example memory configuration:

| | Memory Configuration<br>*Global Bus Width(Data + ECC)* | Frequency | Validation method |
|---|---|---|---|
| Xilinx FPGA<br>*Virtex 5 & Kintex 7* | 8 (8+0) | 333 MHz | Board |
| | 16 (8+5) | 333 MHz | Board |
| | 24 (16 +6) | 333 MHz | Board |
| | 32 (32+0) | 333 MHz | Board |
| | 40 (32 +7) | 333 MHz | Board |
| | 48 (32+16) | 333 MHz | Board |
| | 64 (32+32) | 333 MHz | Simulation |
| | 72 (64+8) | 333 MHz | Simulation |
| | 96 (64+32) | 333 MHz | Simulation |
| Microsemi FPGA<br>*RTG4* | 8 (8+0) | 200MHz | Board |
| | 16 (8+5) | 200MHz | Board |
| | 24 (16 +6) | 200MHz | Board |
| | 32 (32+0) | 200MHz | Board |
| | 40 (32 +7) | 200MHz | Simulation |
| | 48 (32 + 16) | 200MHz | Simulation |

Example configuration of generic nCK_PER_CLK:

| Customer need | RIMC configuration | Memory Controller frequency |
|---|---|---|
| 200 MHz, 400 Mbits/s | nCK_PER_CLK = 4 | 50 MHz |
| 125 MHz, 250 Mbits/s | nCK_PER_CLK = 2 | 62.5 MHz |
| 125 MHz, 250 Mbits/s | nCK_PER_CLK = 4 | 31.25 MHz |

### 6.2 IP SIZE

The following sizes are given for the complete RIMC IP composed of both the Memory Controller and the PHY for a given FPGA target.

Table 31 and Table 32 give the estimated sizes for different FPGA platforms based on the following Hypothesis :

- DDR2 48 bits (32 data + 16 ECC)
- 2 user AMBA interfaces, each one supporting 64 bits
- nCK_PER_CLK = 2 for Kintex 7 and RTG4
- nCK_PER_CLK = 1 for Virtex 5

The precise IP Size depends on the configuration.

| FPGA | Slice registers | Slice LUTs |
|---|---|---|
| Xilinx Virtex 5 | 3570 | 3892 |
| Xilinx Kintex 7 | 8615 | 8217 |

Table 31: Xilinx resource usage

| Type | Used |
|---|---|
| 4LUT | 5000 |
| DFF | 5000 |
| Logic element | 7000 |

Table 32: Microsemi RTG4 resource usage

# 7 DELIVERABLES

The RIMC IP is delivered with the following directory structure:

- 3DIPMC0700-1:
  - source_scr: This directory contains all of the VHDL files for the Memory Controller IP core as well as PHY cores and RIMCs.
    - ip: This directory contains all of the VHDL files of the Memory Controller core. These files are obfuscated.
    - wrapper: This directory contains the VHDL files needed to connect the Memory Controller IP core and the physical layer
    - ddr2_v5_phy: Physical layer for a Xilinx Virtex5 FPGA
    - ddr2_7series_phy: Physical layer for a Xilinx Kintex7 or Virtex7 FPGA
    - ddr2_rtg4_phy: Physical layer for a Microsemi RTG4 FPGA
  - routage: This directory contains Xilinx constraint files used only for examples, since the final constraint file depends on the user pinout.
  - impact_v5: Contains a batch file (implement_example.sh) which can be used to test the synthesis of the RIMC IP for a Virtex 5 FPGA
  - impact_7series: Contains a batch file (implement.sh) which can be used to test the synthesis of the RIMC IP for a Kintex 7 FPGA
  - doc: Contains the datasheet of the RIMC

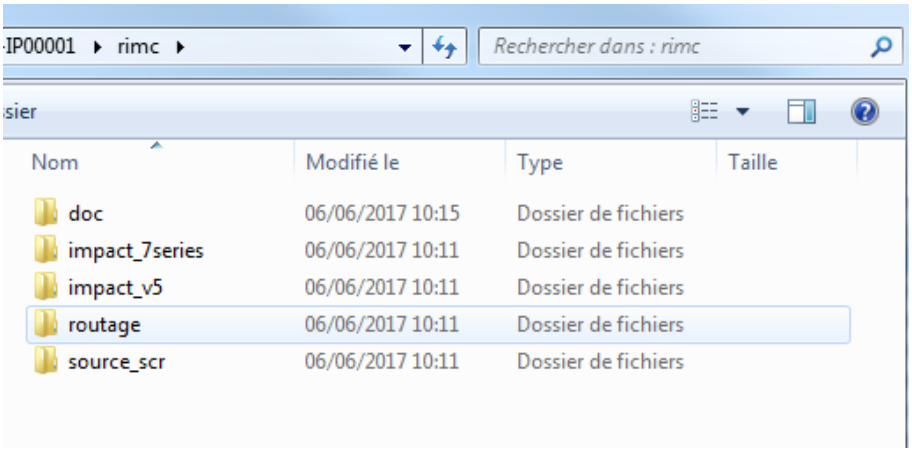



Figure 5: Directory architecture of the deliverables

# 8 TARGET COMPATIBILITY

The RIMC IP core is delivered as high level VHDL source code. It is compatible with a wide range of ASICs and FPGAs.

To date hardware validation has been performed with Xilinx and MicroSemi FPGAs.

## 9 PART NUMBER / ORDER INFORMATION

**3DIPMC0700-1**

**Main Sales Office**

| | | | | | |
|---|---|---|---|---|---|
| **FRANCE** | 3D PLUS<br>408, rue Hélène Boucher ZI.<br>78532 BUC Cedex | Tel : 33 (0)1 30 83 26 50 | Fax : 33 (0)1 39 56 25 89 | Web : www.3d-plus.com<br>e-mail : sales@3d-plus.com | **DISTRIBUTOR** |
| **USA** | 3D PLUS USA, Inc<br>151 Callan Avenue,<br>Suite 310<br>San Leandro, CA 94577 | Tel : (510) 824-5591 | Tel : (510) 824-5591 | e-mail : sales@3d-plus.com | |

# 10 REVISION HISTORY

**Rev. 9, July 2021**

- Remove the Note 2 in paragraph 6.1

**Rev. 8, October 2019**

- Update available and default values for BANK_TYPE generic
- Update Bypass_Reg register
- Update OUTPUT_DRV generic definition
- RTG4: Remove restriction that only IO banks 0 and 9 should be used for RIMC.
- Add SLOT_NUM generic for Memory Controller
- Remove FREQ_RATIO generic

**Rev. 7, January 2019**

- Table 19: Update Scrubbing_Reg0 reset value, and add a note
- Table 24: Add a note regarding CorBeat_Reg register width
- Add ECC_EN and G_RAM_INST generics
- The generic USE_DFF can now have the value 2

**Rev. 6, February 2018**

- Update Figure 1
- Add BA_REPLICATION, ADDR_REPLICATION, CMD_REPLICATION, CLKIN_PERIOD, tCK, CAS_LAT, TWO_T_TIME_EN, TWR_PS, TWTR_PS, TRTP_PS, CLK_WIDTH, DM_WIDTH, DQS_WIDTH, FPGA_SPEED_GRADE generics to RIMC in Table 30
- Add tphy_wrdata, tphy_wrlat, trddata_en, TWR_PS, TWTR_PS, TRTP_PS, CLK_PERIOD, CAS_LAT, TWO_T_TIME_EN, BURST_TYPE generics to memory controller in Table 8
- Add Table 21 to 24 with description of Scrub_Ctl, CorEn_Reg, CorVec_Regx, CorBeat_Reg registers
- Remove memory controller register Table EDACConf_Reg register
- IP size for RTG4 is given with nCK_PER_CLK generic equal to 2

**Rev. 5, October 2017**

- Update syntax on the entire document
- Precise Data width
- Delete memory Redundancy support
- Update Xilinx MIG interface information
- Add details to support Miconsemi RTG4 FPGA
- Delete DDR memory interface subsystem figure
- Update bypass interface signals
- Table 8 updated from RIMC Generic list to Memroy Controller Generic/Parameter list
- Update Chapter5.2 from WRAPPER GENERICS AND PORTS to RIMC GENERICS/PARAMETERS AND PORTS
- Add Chapter6 RIMC Performance

**Rev. 4, June 2017**

- Paragraphs 5 & 6: Introduce RIMC and RIMC+ PHY IP cores in 2 seperated Paragraphs
- Paragraph 4: figure1 updated
- Paragraph 5.5: register list table added and delete DDR_SEL register
- Paragraph 4.2.3: Remove cold redundancy management

- Paragraph 5: Remove APB_DIS generic
- Paragraph 7: IP Size updated

**Rev. 3, Feburary 2017**

- Figure 1 updated to add MIG interface and Bypass interface
- Paragraph 4.1 Initalization pin updated
- Table1 updated with DDR2 component width
- Paragraph 4.2 Data path example was deleted
- Paragraph 6 RIMC Generics and Ports updated
- Paragraph 7 RIMC Registers updated
- Paragraph 8 IP Size added

**Rev. 2, July 2016**

- Initial datasheet release

**Rev. 1, March 2016**

- Initial flyer release