CoreLink™ DDR2 Dynamic Memory Controller (DMC-341)

Revision: r1p1

Technical Reference Manual

ARM®

ARM DDI 0418E (ID080910)

CoreLink DDR2 Dynamic Memory Controller (DMC-341)
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
29 March 2007	A	Non-Confidential	First release for r0p0
16 October 2007	B	Non-Confidential	First release for r0p1
19 December 2007	C	Non-Confidential	Update for r0p1
17 April 2009	D	Non-Confidential	First release for r1p0
05 July 2010	E	Non-Confidential	First release for r1p1

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM® in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means "ARM or any of its subsidiaries as appropriate".

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
CoreLink DDR2 Dynamic Memory Controller (DMC-341) Technical Reference Manual

Preface

Chapter 1 Introduction

Chapter 2 Functional Description

Chapter 3 Programmers Model

Chapter 4 Programmers Model for Test

Appendix A Signal Descriptions

List of Tables
CoreLink DDR2 Dynamic Memory Controller (DMC-341) Technical Reference Manual

List of Figures
CoreLink DDR2 Dynamic Memory Controller (DMC-341) Technical Reference Manual

Preface

This preface introduces the *CoreLink DDR2 Dynamic Memory Controller (DMC-341) Technical Reference Manual*. It contains the following sections:

- *About this book* on page x
- *Feedback* on page xiii.

About this book

This is the *Technical Reference Manual* (TRM) for the CoreLink *DDR2 Dynamic Memory Controller* (DDR2 DMC). The DDR2 DMC comprises various systems that you can render to support a single type and size of DDR2 SDRAM on the memory interface.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n*	Identifies the major revision of the product.
p*n*	Identifies the minor revision or modification status of the product.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) device that uses the DDR2 DMC. The DDR2 DMC provides an interface between the *Advanced eXtensible Interface* (AXI™) system bus and external, off-chip, memory devices.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the DDR2 DMC and its features.

Chapter 2 *Functional Description*

Read this for an overview of the major functional blocks and the operation of the DDR2 DMC.

Chapter 3 *Programmers Model*

Read this for a description of the DDR2 DMC memory map and registers.

Chapter 4 *Programmers Model for Test*

Read this for a description of the DDR2 DMC test registers.

Appendix A *Signal Descriptions*

Read this for a description of the DDR2 DMC signals.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

- *Typographical* on page xi
- *Timing diagrams* on page xi
- *Signals* on page xi.

Typographical

The typographical conventions are:

italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace` Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`monospace` Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic` Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold` Denotes language keywords when used outside example code.

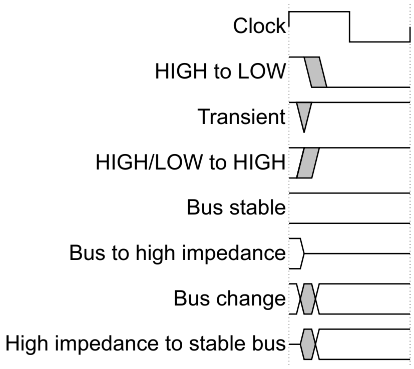
< and > Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
```

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals
- LOW for active-LOW signals.

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

For access to ARM documentation, see Infocenter, `http://infocenter.arm.com`.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *CoreLink DDR2 Dynamic Memory Controller (DMC-341) Implementation Guide* (ARM DII 0183)
- *CoreLink DDR2 Dynamic Memory Controller (DMC-341) Integration Manual* (ARM DII 0184)
- *CoreLink DDR2 Dynamic Memory Controller (DMC-341) Supplement to AMBA® Designer (ADR-301) User Guide* (ARM DSU 0007)
- *AMBA Designer (ADR-301) User Guide* (ARM DUI 0333)
- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *AMBA 3 APB Protocol Specification* (ARM IHI 0024).

Other publications

This section lists relevant documents published by third parties:

- *DDR PHY Interface (DFI) Specification*, `http://www.ddr-phy.org`
- *JEDEC STANDARD DDR2 SDRAM Specification*, JESD79-2, `http://www.jedec.org`.

Feedback

ARM welcomes feedback on the DDR2 DMC and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DDI 0418E
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the CoreLink DDR2 DMC and contains the following sections:

- *About the CoreLink DDR2 DMC (DMC-341)* on page 1-2
- *Product revisions* on page 1-4.

1.1 About the CoreLink DDR2 DMC (DMC-341)

The DDR2 DMC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

The DDR2 DMC is a high-performance, area-optimized DDR2 SDRAM memory controller compatible with the AMBA AXI protocol.

You can configure the DDR2 DMC with a number of options:

- the AXI and DDR2 SDRAM memory data width
- the number of DDR2 SDRAM memory devices
- the number of outstanding AXI addresses
- *Error Correction Code* (ECC) support using *single-error correct, double-error detect* (SECDED)
- the pad interface type, for connection to the *PHYsical* (PHY) device.

For more information about AMBA see:

- *AMBA AXI Protocol Specification*
- *AMBA 3 APB Protocol Specification*.

Figure 1-1 shows an example system.

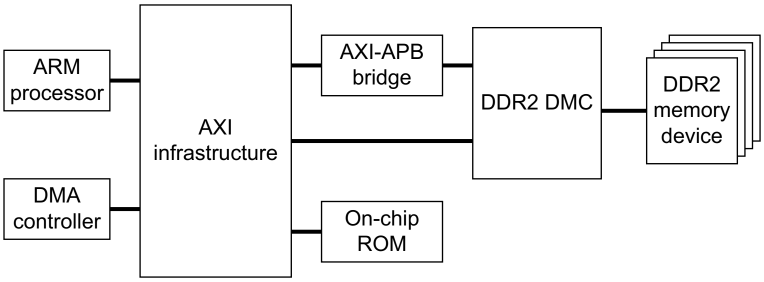


Figure 1-1 Example system

1.1.1 Features of the DDR2 DMC

The DDR2 DMC has the following features:

- compatible with AMBA AXI for accesses to DDR2 memory devices
- compatible with AMBA APB for programming the controller
- soft macrocell available in Verilog
- synchronous n:1 clocking between AXI and APB
- synchronous operation between AXI bus infrastructure and external memory bus using clock ratios of 1:1, 1:n, or n:1
- asynchronous operation between AXI bus infrastructure and external memory bus
- active and precharge power-down supported in the DDR2 SDRAM
- interfaces to a PHY device using either:
  - legacy pad interface
  - *DDR PHY Interface* (DFI) pad interface
- *Quality of Service* (QoS) features for low latency transfers

- optimized utilization of external memory bus
- programmable selection of external memory width, see *Supported memory widths*
- configurable bus width for the **user_status**, **user_config0**, and **user_config1** signals
- multiple outstanding transactions
- write data interleaving supported
- SECDED ECC support
- hardware resource that can be rendered to optimize area versus performance
- support for a configurable number of ARMv6 architecture outstanding exclusive access transfers
- a DDR2 DMC can be configured to have between 1 and 4 memory chip selects.

1.1.2 Supported memory widths

The read data FIFO and write data buffer widths are equal to the greater of the AXI or the effective memory data width.

Table 1-1 shows the supported combinations of memory and AXI bus widths.

Table 1-1 Supported combinations of memory and AXI data widths

Combination	Memory data width	Effective memory data width[a]	AXI data width
a	16-bit	32-bit	32-bit
b	16-bit	32-bit	64-bit
c	32-bit	64-bit	32-bit
d	32-bit	64-bit	64-bit
e	32-bit	64-bit	128-bit
f	64-bit	128-bit	64-bit
g	64-bit	128-bit	128-bit

[a] Effective memory data width is equal to the size of transfer on a per-cycle basis on the memory interface.

Note

- In addition to the choice of memory data widths at render time, you can modify the DDR2 DMC to use half the configured memory width by either:
  - — using the **memory_width[1:0]** tie-off signals
  - — programming the memory_width2 field in the *Memory Configuration 2 Register* on page 3-22.
- When modifying the configured memory width you must ensure that the:
  - — new memory width is not less than 16 bits
  - — effective memory width is not less than half the AXI slave interface data width.

1.1.3 Supported memory devices

See the product *Release Note* for more information.

1.2 Product revisions

This section describes the differences in functionality between product revisions of the DDR2 DMC:

r0p0-r0p1 Contains the following differences in functionality:

- the revision field returns `0x1` or `0x2`, see *Peripheral Identification Register 2* on page 3-36.

r0p1-r1p0 This release includes:

- the option to configure the DDR2 DMC to support:
  - a DFI pad interface, see *Pad interface* on page 2-25
  - ECC, see *ECC block* on page 2-19
- the controller can enter the Config state from the Paused state, irrespective of which command moved the controller to the Paused state, see *Memory manager* on page 2-17
- supports memory bursts of four and eight, see *Formatting from AXI address channels* on page 2-11
- configurable bus width for the **user_status**, **user_config0**, and **user_config1** signals, see *User signals* on page A-4
- the revision field returns `0x3`, see *Peripheral Identification Register 2* on page 3-36.

r1p0-r1p1 This release includes:

- the revision field returns `0x4`, see *Peripheral Identification Register 2* on page 3-36.

Chapter 2
Functional Description

This chapter describes the CoreLink DDR2 DMC operation. It contains the following sections:

- *Functional overview* on page 2-2
- *Functional operation* on page 2-8.

2.1 Functional overview

Figure 2-1 shows the interfaces of the DDR2 DMC.

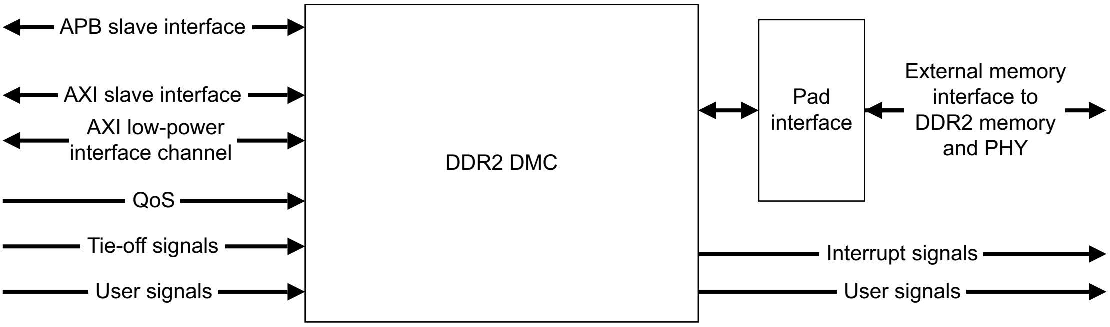


Figure 2-1 DDR2 DMC interfaces

The interfaces of the DDR2 DMC are:

- *AXI slave interface*
- *AXI low-power interface* on page 2-4
- *APB slave interface* on page 2-4
- *Tie-off signals* on page 2-4
- *User signals* on page 2-4
- *Interrupt signals* on page 2-5
- *Memory interface* on page 2-5
- *Pad interface* on page 2-6
- *QoS signal* on page 2-7.

2.1.1 AXI slave interface

The AXI slave interface comprises the following AXI channels:

Write address channel

Enables the transfer of the address and all other control data required for the DDR2 DMC to carry out an AXI write transaction.

Write data channel

Enables the transfer of write data and validating data byte strobes to the controller.

Write response channel

Enables the transfer of response information associated with a write transaction.

Read address channel

Enables the transfer of the address and all other control data required for the controller to carry out an AXI read transaction.

Read data channel Enables the transfer of read data and response information associated with a read transaction.

See the *AMBA AXI Protocol Specification* for more information.

Figure 2-2 on page 2-3 shows the AXI slave interface signals.

Note

In Figure 2-2 the **arcache**, **awcache**, **arprot**, and **awprot** signals are shown for completeness only. The controller ignores any information that these signals provide.

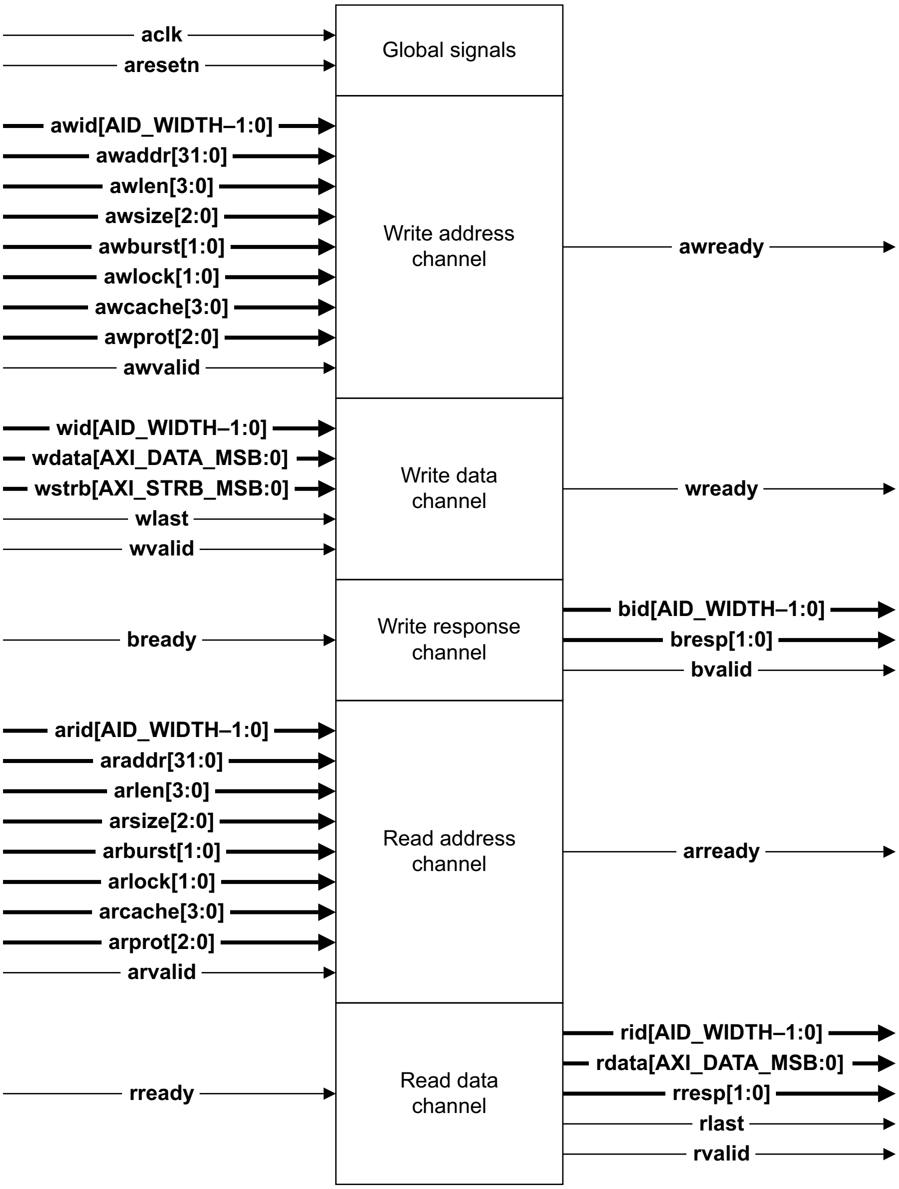


Figure 2-2 AXI slave interface signals

2.1.2 AXI low-power interface

Figure 2-3 shows the AXI low-power interface channel signals.

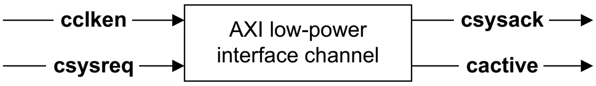


Figure 2-3 AXI low-power interface channel signals

See *AXI low-power interface* on page 2-13 for more information.

2.1.3 APB slave interface

The APB slave interface provides access to the internal registers of the DDR2 DMC. Figure 2-4 shows the APB interface signals.

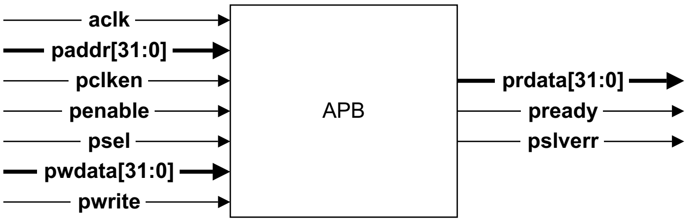


Figure 2-4 APB interface

See *APB slave interface* on page 2-13 for more information.

2.1.4 Tie-off signals

The tie-off signals initialize various operating parameters of the DDR2 DMC when it exits the reset state. Figure 2-5 shows the tie-off signals.

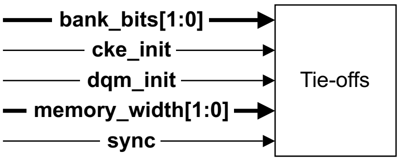


Figure 2-5 Tie-off signals

——— **Note** ———

cke_init and **dqm_init** are only available if the DDR2 DMC is configured to contain a legacy pad interface.

———————

See *Tie-off signals* on page 2-14 for more information.

2.1.5 User signals

The user signals are general purpose input and output signals that you can control and monitor by using the registers that the DDR2 DMC provides. Figure 2-6 on page 2-5 shows the user signals.

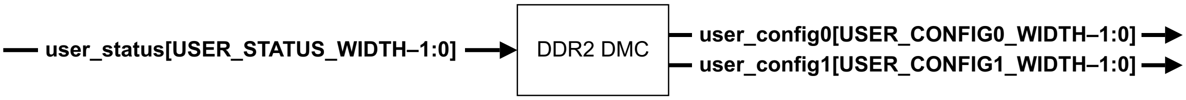

user_status[USER_STATUS_WIDTH–1:0]
DDR2 DMC
user_config0[USER_CONFIG0_WIDTH–1:0]
user_config1[USER_CONFIG1_WIDTH–1:0]

Figure 2-6 User signals

See *Miscellaneous signals* on page 2-14 for more information.

2.1.6 Interrupt signals

Figure 2-7 shows the interrupt signals that the DDR2 DMC provides if it is configured to support *Error Correction Code* (ECC).

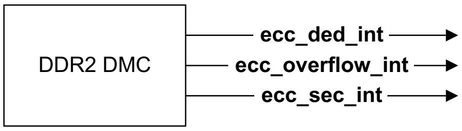


Figure 2-7 Interrupt signals

See *ECC block* on page 2-19 for more information about how the ECC block controls these signals.

2.1.7 Memory interface

The memory interface provides a clean and defined interface between the pad interface and the arbiter, ensuring that the external memory interface command protocols are met in accordance with the programmed timings in the register block. See Chapter 3 *Programmers Model*.

The external inputs and outputs to this block are:

mclk Clock for **mclk** domain.

mresetn Reset for **mclk** domain. This signal is active LOW.

The memory interface tracks and controls the state of an external memory by using an **mclk** FSM. See Figure 2-8.

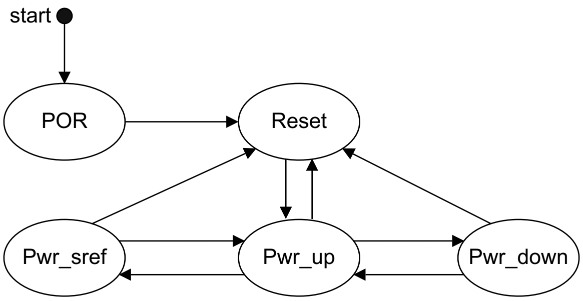


Figure 2-8 mclk domain state diagram

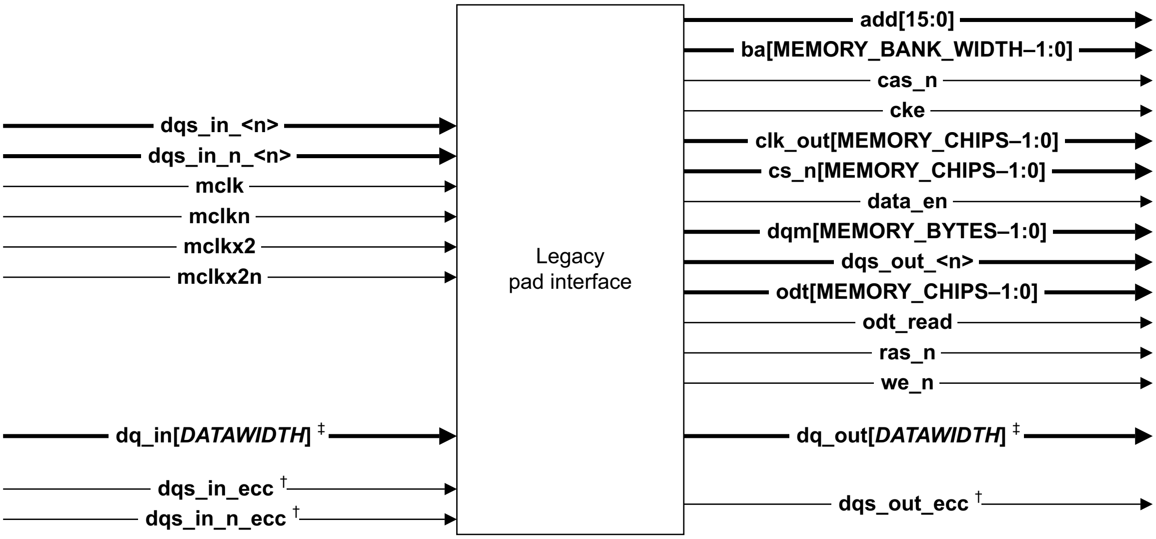
See Table 2-5 on page 2-29 for valid system states.

See *Memory interface* on page 2-20 for more information.

2.1.8 Pad interface

You can configure the DDR2 DMC to provide either a:

- legacy pad interface, see Figure 2-9
- DFI pad interface, see Figure 2-10.

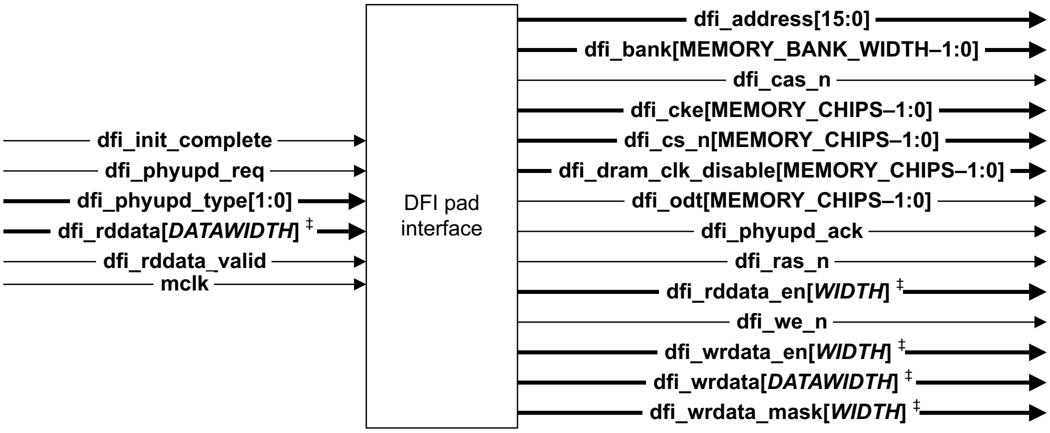


† This signal is only available when a DDR2 DMC supports ECC.

‡ If a DDR2 DMC supports ECC then:
- *DATAWIDTH* = **(MEMWIDTH+ECCWIDTH)–1:0**

otherwise:
- *DATAWIDTH* = **MEMWIDTH–1:0**

Figure 2-9 Legacy pad interface signals



‡ If an DDR2 DMC supports ECC then:
- *DATAWIDTH* = **2×(MEMWIDTH+ECCWIDTH)–1:0**
- *WIDTH* = **2×(MEMORY_BYTES+1)–1:0**

otherwise:
- *DATAWIDTH* = **2×MEMWIDTH–1:0**
- *WIDTH* = **2×MEMORY_BYTES–1:0**

Figure 2-10 DFI pad interface signals

See *Pad interface* on page 2-25 for more information.

2.1.9 QoS signal

An AMBA master can use the QoS signal to request that the DDR2 DMC assigns the minimum latency to the current read transaction. Figure 2-11 shows the QoS signal.

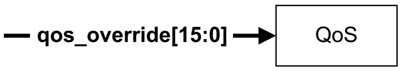


Figure 2-11 QoS signal

See *Quality of Service* on page 2-16 for more information.

2.2 Functional operation

Figure 2-12 shows a block diagram of the DDR2 DMC.

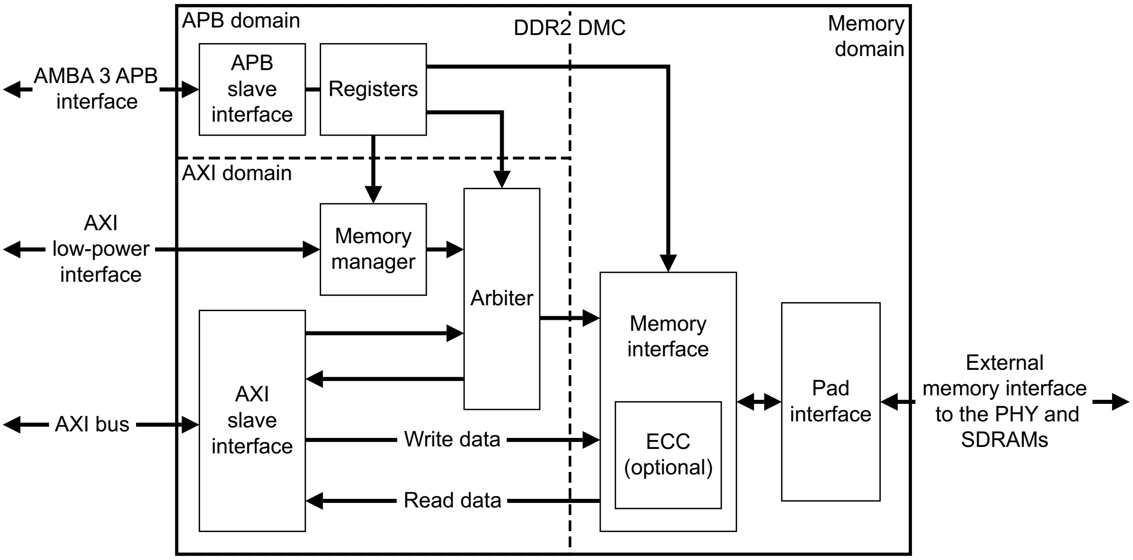


Figure 2-12 Block diagram

This section describes:

- *Clocking and resets*
- *AXI slave interface* on page 2-9
- *AXI low-power interface* on page 2-13
- *APB slave interface* on page 2-13
- *Tie-off signals* on page 2-14
- *Miscellaneous signals* on page 2-14
- *Arbiter* on page 2-14
- *Memory manager* on page 2-17
- *ECC block* on page 2-19
- *Memory interface* on page 2-20
- *Pad interface* on page 2-25
- *Initialization* on page 2-26
- *Power-down support and usage model* on page 2-28.

2.2.1 Clocking and resets

This section describes:

- *Clocking*
- *Reset* on page 2-9.

Clocking

The DDR2 DMC has the following functional clock inputs:

- **aclk**
- **mclk**
- **mclkn**
- **mclkx2**

- **mclkx2n**
- **dqs_in_<n>** where n is 0<n<bytes of external memory data bus
- **dqs_in_n_<n>** where n is 0<n<bytes of external memory data bus.

——— **Note** ———

mclkn, **mclkx2**, **mclkx2n**, **dqs_in_<n>**, and **dqs_in_n_<n>** are only available if the controller is configured to contain a legacy pad interface.

———

These clocks can be grouped into two clock domains:

aclk domain **aclk** is in this domain. The **aclk** domain signals can only be stopped if the external memories are put in self-refresh mode.

mclk domain All clocks except **aclk** are in this domain. The **mclk** signal must be clocked at the rate of the external memory clock speed. The **mclk** domain signals can only be stopped if the external memories are put in self-refresh mode.

——— **Note** ———

If the controller contains a legacy pad interface then it provides a separate clock output for every memory device. If the controller contains a DFI pad interface then it asserts **dfi_dram_clk_disable[MEMORY_CHIPS–1:0]**, to the DFI PHY, to disable the clock to a memory device.

———

Reset

The DDR2 DMC has two reset inputs:

aresetn This is the reset signal for the **aclk** domain.

mresetn This is the reset signal for the **mclk** domain.

You can change both reset signals asynchronously to their respective clock domain. Internally to the controller, the deassertion of the **aresetn** signal is synchronized to **aclk**, and the deassertion of the **mresetn** signal is synchronized to:

- **mclk** for a DDR2 DMC configured with DFI
- the **mclk**, **mclkn**, **mclkx2**, and **mclkx2n** clock signals for a DDR2 DMC configured with a legacy pad interface.

2.2.2 AXI slave interface

The AXI programmer's view is of a flat area of memory. The full range of AXI operations are supported.

The base addresses of the external memory devices are programmable using the chip_cfg<n> Registers. See *Chip Configuration Registers* on page 3-28.

In addition to reads and writes, exclusive reads and writes are supported in accordance with the *AMBA AXI Protocol Specification*. Successful exclusive accesses have an EXOKAY response. All other accesses, including exclusive fail accesses, receive an OKAY response.

Note

Refreshes can be missed if **rready** is held LOW for longer than one refresh period and the read data FIFO, command FIFO, and arbiter queue become full. An OVL error is triggered if this occurs in simulation. Ensure that the DDR2 DMC has a sufficiently high system priority to prevent this.

This section describes:

- *Configuration options*
- *Write data merging*
- *Early BRESP*.

Configuration options

You can choose 128-bit, 64-bit, or 32-bit data width.

Write data merging

The DDR2 DMC merges interleaved write data to optimize the utilization of the memory interface.

Early BRESP

To enable early write response timing, the DDR2 DMC employs write data buffering and issues the **BRESP** transfer before the data has been committed to the DDR2 SDRAM. The response is sent after the last data beat is accepted by the AXI slave interface and stored in the write data buffer. You can disable this feature using the *Feature Control Register* on page 3-31.

Note

For exclusive write accesses, the controller only issues a **BRESP** transfer after the write transaction is committed to a memory device.

AXI slave interface attributes

Table 2-1 shows the AXI slave attributes and their values.

Table 2-1 AXI slave interface attributes

Attribute[a]	Value
Combined acceptance capability	Arbiter queue depth
Write interleave depth	Combined acceptance capability
Read data reorder depth	Combined acceptance capability

[a] See *Glossary* for a description of these AXI attributes.

Formatting from AXI address channels

Formatting is as follows:

Chip select decoding

Using the programmed values in the chip_cfg<n> Registers, see *Chip Configuration Registers* on page 3-28, that Chapter 3 *Programmers Model* defines, an incoming address has the most significant eight address bits compared with the address match bits using the address mask to ignore any don't care bits to select an external chip.

The transfer is still carried out if there is no match, but the result is undefined.

Row select decoding

The row address is determined from the AXI address using bits [5:3] of the memory_cfg Register, and also the brc_n_rbc bit for the selected chip defined in the chip_cfg<n> Register. See *Memory Configuration Register* on page 3-12 and *Chip Configuration Registers* on page 3-28.

Column select decoding

The column address is determined from the AXI address using bits [2:0] of the memory_cfg Register. See *Memory Configuration Register* on page 3-12.

Bank select decoding

The chip bank is determined from the AXI address using bits [5:3] of the memory_cfg2 Register, and also the brc_n_rbc bit for the selected chip defined in the chip_cfg<n> Register. See *Memory Configuration 2 Register* on page 3-22 and *Chip Configuration Registers* on page 3-28.

Number of beats

The number of memory beats is determined, depending on the effective external memory width and the burst size of the AXI access. AXI wrapping bursts are split into two incrementing bursts. AXI fixed bursts are split into AXI burst length memory bursts.

Note

On the memory interface, the DDR2 DMC supports a burst length of four or eight.

Exclusive access

In addition to reads and writes, the DDR2 DMC supports exclusive reads and writes in accordance with the *AMBA AXI Protocol Specification*.

Successful exclusive accesses have an EXOKAY response. All other accesses, including exclusive fail accesses, receive an OKAY response.

Exclusive access monitors implement the exclusive access functionality. Each monitor can track a single exclusive access. The number of monitors is a configurable option.

If an exclusive write fails, the data mask for the write is forced LOW, so the data is not written.

When monitoring an exclusive access, the address of any write from another master is compared with the monitored address to check that the location is not being updated.

For the purposes of monitoring, address comparison uses a bit mask derived as follows:

Consider the byte addresses accessed by a transaction. All of the least significant bits up to and including the most significant bit that vary between those addresses are set to logic zero in the mask. All of the stable address bits above this point are set to `0b1`.

Example 2-1 provides information about three transactions.

Example 2-1 Exclusive accesses

Exclusive Read	Address = `0x000`, size = WORD, length = 1, ID = 0.
Write	Address = `0x004`, size = WORD, length = 2, ID = 1.
Exclusive Write	Address = `0x000`, size = WORD, length = 1, ID = 0.

The write transaction accesses the address range `0x104`-`0x10B` and address bit 3 is the most significant bit that varies between byte addresses. Because address bits 3 down to 0 are not compared, the monitoring logic calculates as if the masked write has accessed the monitored address and marks the exclusive write as a failure.

Table 2-2 shows the address comparison steps.

Table 2-2 Address comparison steps example

Step		Binary	Hex
1	Monitored address	`0b000100000000`	`0x100`
2	Write address	`0b000100000100`	`0x104`
3	Write accesses	`0b000100000100`	`0x104`
		`0b000100000101`	`0x105`
		`0b000100000110`	`0x106`
		`0b000100000111`	`0x107`
		`0b000100001000`	`0x108`
		`0b000100001001`	`0x109`
		`0b000100001010`	`0x10A`
		`0b000100001011`	`0x10B`
4	Generate a comparison mask	`0b111111110000`	`0xFF0`
5	Monitored address ANDed with mask	`0b000100000000`	`0x100`
6	Write address ANDed with mask	`0b000100000000`	`0x100`
7	Compare steps 5 and 6		
8	Mark exclusive write as failed		

This example shows how the logic can introduce false negatives in exclusive access monitoring, because in reality the write does not access the monitored address. The implementation is chosen to reduce design complexity but always provide safe behavior.

When calculating the address region accessed by the write, the burst type is always taken to be INCR. Therefore, a wrapped transaction in Example 2-1 on page 2-12 that wraps down to `0x0` rather than cross the boundary, is treated in the same way. This is the same for a fixed burst that does not cross the boundary or wrap down to `0x0`.

2.2.3 AXI low-power interface

The low-power interface can move the DDR2 DMC into its Low_power state without the requirement for any register accesses, see *aclk domain state diagram* on page 2-18 and *Power-down support and usage model* on page 2-28.

For more information about the AXI low-power interface, see the *AMBA AXI Protocol Specification*.

If you do not require the low-power interface, tie it off as the *CoreLink DDR2 Dynamic Memory Controller (DMC-341) Integration Manual* describes.

2.2.4 APB slave interface

The APB slave interface is a fully compliant APB slave. See the *AMBA 3 APB Protocol Specification* for information about an APB slave interface.

The APB interface enables you to access the operating state of the DDR2 DMC and to program it with the correct timings and settings for the connected memory type. See Chapter 3 *Programmers Model* for more information. The APB interface also initializes the connected memory devices, see *Initialization* on page 2-26.

Note

The APB only supports single-word 32-bit accesses. The DDR2 DMC ignores **paddr[1:0]**, resulting in byte and halfword accesses being treated as word accesses.

The APB interface is clocked by the same clock as the AXI domain clock, **aclk**. The controller provides a clock enable, **pclken**, enabling the APB interface to be slowed down and execute at an integer divisor of **aclk**.

To enable a clean registered interface to the external infrastructure, the APB interface always adds a wait state for all reads and writes by driving **pready** LOW. In the following instances, a delay of more than one wait state can be generated when a:

- direct command is received and there are outstanding commands that prevent a new command being stored in the command FIFO
- memory command is received, and a previous memory command has not completed.

The only registers that can be accessed when the controller is not in the Config or Low_power state are:

- memc_status Register, to read the current state, see *Memory Controller Status Register* on page 3-8
- memc_cmd Register, to change state, see *Memory Controller Command Register* on page 3-9.

To guarantee no missed `AUTO REFRESH` commands, it is recommended that any change of **mclk** period, and therefore update of the refresh period, is carried out when the controller is in the Low_power state. This is because the refresh rate depends on the **mclk** period. Only write direct commands to the external memories when the controller is in the Config state and not in the Low_power state.

2.2.5 Tie-off signals

The DDR2 DMC enables you to change some of its configuration settings by using the tie-off signals.

At reset, the value of each tie-off signal controls the respective bits in the memory_cfg2 Register.

After reset you can program the memory_cfg2 Register to make additional changes to these configuration settings. See *Memory Configuration 2 Register* on page 3-22.

2.2.6 Miscellaneous signals

You can use the following general-purpose signals for controlling or monitoring logic that is external to the DDR2 DMC:

user_status[USER_STATUS_WIDTH–1:0]

You can read the status of these general-purpose inputs by using the *User Status Register* on page 3-29. You must tie any unused signals to either HIGH or LOW. These signals are connected directly to the APB interface block. Therefore, if they are driven from external logic that is not clocked by the **aclk** signal, then external synchronization registers are required.

user_config0[USER_CONFIG0_WIDTH–1:0]

The user_config0 Register controls these general-purpose outputs, see *User Config 0 Register* on page 3-30. If you do not require these signals, leave them unconnected.

user_config1[USER_CONFIG1_WIDTH–1:0]

The user_config1 Register controls these general-purpose outputs, see *User Config 1 Register* on page 3-30. If you do not require these signals, leave them unconnected.

You can use the following miscellaneous signals, as tie-offs, to change the operational behavior of the controller:

rst_bypass Use this signal for *Automatic Test Pattern Generation* (ATPG) testing only. You must tie it LOW for normal operation.

dft_en_clk_out

Use this signal for ATPG testing only. It is only available when the controller is configured to provide a legacy pad interface. You must tie it LOW for normal operation.

2.2.7 Arbiter

The arbiter receives memory access commands from the AXI slave interface and the memory manager. It passes the highest priority command to the memory interface after arbitration. Read data is passed from the memory interface to the AXI slave interface.

This section describes:

- *Formatting from memory manager* on page 2-15
- *Arbiter access multiplexor* on page 2-15
- *Hazard detection* on page 2-15
- *Scheduler* on page 2-16
- *Quality of Service* on page 2-16

- *Arbitration algorithm* on page 2-17
- *Command formatting* on page 2-17.

Formatting from memory manager

The direct command uses the chip select bits [21:20] in the direct_cmd Register to select the required memory chip. See *Direct Command Register* on page 3-11.

The command to be carried out is either:

- a self-refresh request from the AXI-C interface
- an auto-refresh
- a direct command from the APB interface.

It is encoded by the memory manager to match the format that the arbiter requires.

Arbiter access multiplexor

The selection of a command from the AXI slave interface or the memory manager is fixed, with the memory manager having a higher priority.

The selection between an AXI read access and AXI write access is made using a round-robin arbitration, unless a read access has a low latency QoS value, or if any of the accesses are to a row already open. See *Arbitration algorithm* on page 2-17 for more information.

See also *Formatting from AXI address channels* on page 2-11. In this case, it is arbitrated immediately.

Hazard detection

The following types of hazard exist:

Read After Read **(RAR)**

There is a read already in the arbiter queue with the same ID as the incoming entry, that is also a read.

Write After Write **(WAW)**

There is a write already in the arbiter queue with the same ID as the incoming entry, that is also a write.

Read After Write **(RAW)**

There is a write in the arbiter queue, that has received an early write response, accessing the same location as the incoming read entry.

The arbiter entry is flagged as having a dependency if a hazard is detected. There might be dependencies against a number of other arbiter entries. As the arbiter entries are invalidated, so the dependencies are reduced until finally, there are no outstanding dependencies, and the entry is free to start.

Note

There are no *Write-After-Read* (WAR) hazard checks in the DDR2 DMC. If an AXI master requires ordering between reads and writes to certain memory locations, it must wait for read data before issuing a write to a location it has read from. Similarly, the only RAW hazard checking is that performed when the write response has been issued. If an AXI master required ordering between writes and reads to certain memory locations, it must wait for the write response before issuing the read to the same location.

Scheduler

The scheduler monitors the activity of the **mclk** FSMs in the memory interface. This enables the arbiter to select an entry from the queue that does not stall the memory pipeline.

Quality of Service

Quality of Service (QoS) is defined for the DDR2 DMC as a method of increasing the arbitration priority of a read access that requires low-latency read data. The QoS for an AXI read access is determined when it is received by the arbiter.

There is no QoS for write accesses. However, any write access that is a dependency for a QoS read access receives a promoted priority to complete as soon as possible. Dependencies are formed based on the hazard detection logic that *Hazard detection* on page 2-15 describes.

The following sections describe:

- *QoS selection*
- *QoS timeout*
- *QoS for AUTO REFRESH* on page 2-17.

QoS selection

The DDR2 DMC selects the QoS for an AXI read transfer by masking the **arid** with a 4-bit QoS mask. You can program the QoS mask to be either **arid[3:0]**, **arid[4:1]**, **arid[5:2]**, **arid[6:3]**, **arid[7:4]**, **arid[8:5]**, **arid[9:6]**, or **arid[10:7]** by using the qos_master_bits in the memory_cfg Register. After the controller applies the QoS mask to the four specified **arid** bits, the resulting value <n> provides the pointer to which id_<n>_cfg Register contains the QoS settings for the read transfer. See *Memory Configuration Register* on page 3-12 and *QoS Configuration Registers* on page 3-27.

Example 2-2 shows QoS selection and the impact of the **qos_override** signal.

Example 2-2 QoS selection and qos_override

If you program the qos_master_bits = `0b010` then this selects **arid[5:2]** to be the 4-bit QoS mask. If the DDR2 DMC receives an AXI transfer with an **arid** of `0x5A` then it applies the 4-bit QoS mask, **arid[5:2]**, giving a value of `0x6`. Therefore, the controller uses the id_6_cfg Register to control the QoS for the transfer.

The controller creates a new arbiter entry for the transfer and assigns it the qos_min and qos_max values from the id_6_cfg Register. If the qos_enable bit=1 then the controller applies the QoS settings to the transfer.

The **qos_override[15:0]** signal enables the controller to assign an arbiter entry with minimum QoS latency, irrespective of the state of the qos_enable bit. For this example, if **qos_override[6]** is HIGH when **arvalid** and **arready** are HIGH then the arbiter entry is assigned minimum QoS latency, even if the qos_enable bit=0.

QoS timeout

If the qos_enable bit is set then the arbiter decrements the QoS maximum latency value every **aclk** cycle until it reaches zero. If the entry is still in the queue when the QoS maximum latency value reaches zero, then the entry becomes high priority. This is called a *timeout*. Also, any entry in the queue with a minimum latency QoS also produces a timeout. Minimum latency timeouts have priority over maximum latency timeouts.

When an entry times out in this way it forces a timeout onto any entries that it has dependencies against. In normal operation, these entries have already timed out because they have received the same initial QoS value but been decrementing for longer. The highest priority arbiter entry is serviced next.

One special case exists. This is when or if the assertion of the relevant **qos_override** signal forces a minimum latency timeout. In this instance, any accesses that the new entry has dependencies against might not have timed out and are forced to time out so that the high-priority entry can start as soon as possible.

QoS for `AUTO REFRESH`

The DDR2 DMC provides QoS for the `AUTO REFRESH` commands by using a simple increment-decrement counter for each memory device. The counter tracks the number of `AUTO REFRESH` commands in the arbiter queue and when it reaches a value of six then a refresh timeout is signaled to the arbiter queue.

A refresh timeout forces all of the `AUTO REFRESH` queue entries to timeout. This timeout is sticky, and does not disappear when the number of timeouts drops below six. Instead, it remains asserted until the controller services all of the `AUTO REFRESH` entries. This provides a guaranteed refresh rate in the SDRAM.

Arbitration algorithm

The ordering of commands to be carried out from the arbiter queue is arbitrated with a priority scheme of the following order:

- refresh timeouts
- minimum latency timeouts
- maximum latency timeouts
- open-row hits in the same direction
- open-row hits in the opposite direction
- preparation operations
- refreshes.

Note

Preparation operations can be interleaved between memory operations to other banks to improve memory interface utilization.

Command formatting

For every memory burst access necessary to complete an arbiter queue entry, a memory interface command is required.

Command formatting calculates the number of memory interface commands and memory cycles for each command to complete the next arbiter queue entry that is to be sent to the memory interface. It contains an address incrementer and a beat decrementer so that the arbiter entry can be interrupted and restarted.

2.2.8 Memory manager

The memory manager tracks and controls the current state of the DDR2 DMC using the **aclk** *Finite State Machine* (FSM). You can change the state of the controller by programming the memc_cmd Register, see *Memory Controller Command Register* on page 3-9.

You can also use the AXI low-power interface to move the controller between the Ready and Low_power states, see *Power-down support and usage model* on page 2-28.

Figure 2-13 shows the **aclk** FSM.

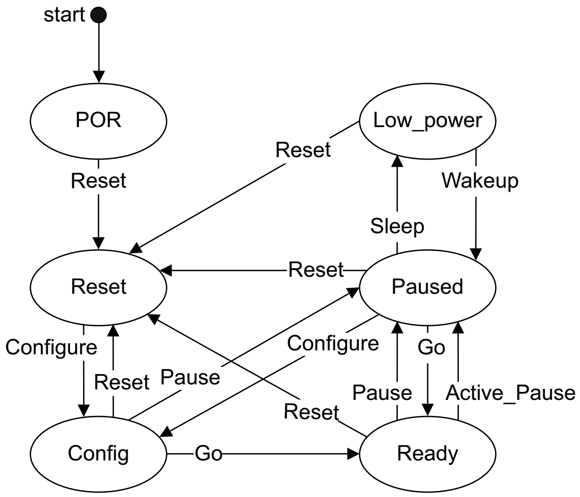


Figure 2-13 aclk domain state diagram

In Figure 2-13, non-state moving transitions are omitted for clarity. See Table 2-5 on page 2-29 for valid system states.

Note

- If the controller receives an APB command that is illegal to carry out from the current state, then the controller ignores it and the **aclk** FSM stays in the same state.
- For the two cycles following *Power-On-Reset* (POR), do not consider the controller to be in the Config state. For this reason, register access restrictions apply.
- You can only use the AXI low-power interface to move in and out of the Low_power state from the Ready state.
- If the controller enters the Low_power state using the:
  - APB interface then it must also exit the Low_power state using the APB interface
  - AXI low-power interface then it must also exit the Low_power state using the AXI low-power interface.
- When the controller is in the Ready state it periodically issues `AUTO REFRESH` commands to the SDRAMs. When the controller is in the Config or Paused state it does not issue `AUTO REFRESH` commands unless software writes the appropriate command to the direct_cmd Register. However, this register is only accessible in the Config state and therefore software must limit the duration of the Paused state.
- If you move the controller from the Low_power state to the Paused state then you must not move the controller immediately back to the Low_power state if it has not sent an `AUTO REFRESH` command. This is because, when a memory device exits self-refresh and then re-enters self-refresh mode, the *JEDEC STANDARD DDR2 SDRAM Specification* requires a controller to issue at least one `AUTO REFRESH` command.

The APB slave interface stalls the **psel** and **penable** signals using the **pready** signal if a previous command has not completed.

The memory manager issues commands from one of the following sources:

Direct commands

These are received over the APB interface as a result of a write to the direct_cmd Register, see *Direct Command Register* on page 3-11. Use these commands to initialize the DDR2 SDRAM and generate periodic refresh commands.

The legal commands that the memory manager uses are:

- `NOP`
- `PRECHARGEALL`
- `AUTO REFRESH`
- `MODEREG`
- extended `MODEREG`.

Commands from the aclk FSM

You can traverse the **aclk** FSM by writing to the memc_cmd Register. See *Memory Controller Command Register* on page 3-9. You can only traverse the **aclk** FSM states when the DDR2 DMC is idle. For example, the Ready state can only be entered from the Config state when all direct commands have been completed. The eption to this is the Active_Pause command. You can issue this command when the controller is active. When you issue the command, any memory accesses that have not been arbitrated remain in the arbiter until the **aclk** FSM receives a Go command.

Refresh commands

The refresh logic can issue commands to the arbiter to refresh the DDR2 SDRAMs. The refresh rate period is programmable using the *Refresh Period Register* on page 3-14. The value of this register is the time period in **mclk** cycles that must occur before the memory manager requests the arbiter to generate an `AUTO REFRESH` command. This request is arbitrated and might not necessarily be initiated immediately. See *Arbiter* on page 2-14.

2.2.9 ECC block

The DDR2 DMC can be configured to include ECC support.

The ECC mechanism uses a *Hamming* code variant, using 1-bit correction 2-bit detection. Program the ecc_control Register to enable the controller to use the ECC block. See *ECC Control Register* on page 3-32.

When ECC is enabled, the controller requires writes:

- to be an integer multiple of the data bus width of a memory device, that is, MEMWIDTH bits
- to be aligned to the data bus width of a memory device
- to assert all of the write strobe signals, that is, sparse data transfers do not occur.

If these conditions are violated then the controller writes invalid ECC data.

When the controller performs a `READ` command, if it detects:

- a single bit-error then it sets **ecc_sec_int** HIGH and corrects the read data
- two or more bit-errors it sets **ecc_ded_int** HIGH but does not correct the data.

Note

The controller can only assert **ecc_sec_int** or **ecc_ded_int** when the corresponding interrupt enable bit is set in the ecc_control Register. See *ECC Information Register* on page 3-34.

If the controller is signaling a single interrupt then an AMBA master can discover the address location that triggered an interrupt by accessing the ecc_info0 Register. See *ECC Information Register* on page 3-34.

The controller provides the **ecc_overflow_int** signal to indicate when an interrupt overflow occurs, that is:

- If an AMBA master fails to clear the **ecc_sec_int** interrupt before the controller corrects another single bit-error then the controller sets **ecc_overflow_int** HIGH.
- Similarly, if an AMBA master fails to clear the **ecc_ded_int** interrupt before the controller detects another multiple bit-error then the controller sets **ecc_overflow_int** HIGH.

Note

If the controller detects two or more errors within the same aligned AXI-width region of a burst then it does not set the overflow bit and **ecc_overflow_int** remains LOW. However, it still performs error correction and detection on each memory width portion of the burst.

The controller controls the enabling and disabling of the **ecc_overflow_int** signal as follows:

- if **ecc_sec_int** and **ecc_ded_int** are disabled then **ecc_overflow_int** is disabled
- if **ecc_sec_int** or **ecc_ded_int** are enabled then **ecc_overflow_int** is enabled.

2.2.10 Memory interface

The memory interface of the DDR2 DMC is separated from the arbiter using the following configurable blocks:

- command FIFO
- read data FIFO
- write data buffer.

The memory interface reads commands from the arbiter using the command FIFO but only when that command can be executed. The memory interface ensures a command is only executed when all the inter-command delays, defined in this section, for that bank or chip are met.

The memory interface enables multiple banks to be active at any one time. However, only one bank can be carrying out a data transfer at any one time. If the command at the head of the command FIFO cannot be executed, then the command pipeline stalls until it can be executed.

When the auto_power_down bit is set, see *Memory Configuration Register* on page 3-12, then the **cke** output signal is negated to take the external memories into active or precharge power-down depending on whether there is a row open. See Figure 2-23 on page 2-23. When exiting power down mode, the delay before the next command is issued is programmed using the *Exit Power-down Timing Register* on page 3-20.

An **mclk** FSM controls the operation of the power-down mode. This FSM has a state that is entered when the DDR2 SDRAM is put into self-refresh mode. This is used so that if power is removed from all of the DDR2 DMC apart from the memory interface and pad interface, the state of the memory is known. When the rest of the controller is powered-up, the **aclk** FSM enters the Low_power state rather than the Config state.

Memory interface to pad interface timing

All command control outputs are clocked on the same edge of the memory clock, **mclk**.

The relative times between control signals from the memory interface are maintained when output from the pad interface to the actual DDR2 memory devices. Therefore, the timing register values required for a particular DDR2 SDRAM can be determined from that memory device's data sheet.

Figure 2-14 to Figure 2-26 on page 2-25 show how the data sheet timings map onto the DDR2 DMC timing registers. The timing registers are shown in Figure 3-2 on page 3-3.

——— **Note** ———

In Figure 2-14 to Figure 2-26 on page 2-25:

- The following signals are internal to the DDR2 DMC:
  - **command_en**
  - **data_cntl_en**
  - **memif_busy**
  - **pwr_down**
  - **read_en**.
- The control outputs to the external memory are always clocked on the falling edge of the memory clock, **mclk**.
- The timings shown are not necessarily the default timing values but are values that are small enough to show the entire delay in one figure.

Figure 2-14 shows the command control output timing.

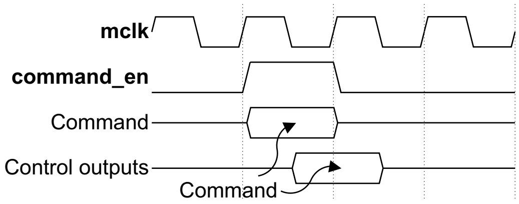


Figure 2-14 Command control output timing

Figure 2-15 shows the `ACTIVE` command to `Read` or `Write` command timing, that you program using the *ACTIVE to Read or Write Timing Register* on page 3-17.

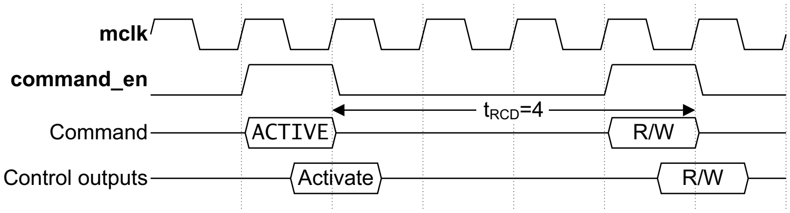


Figure 2-15 `ACTIVE` command to `Read` or `Write` command timing, t_{RCD}

Figure 2-16 on page 2-22 shows the four activate window command timing, that you program using the *Four Activate Window Timing Register* on page 3-24.

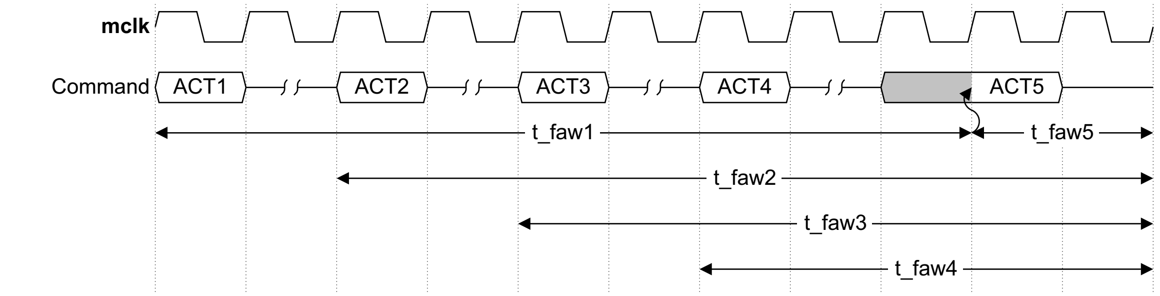


Figure 2-16 Four activate window command timing, t_{FAW}

Figure 2-17 shows ACTIVE to ACTIVE on the same bank, and ACTIVE to AUTO REFRESH command timing, that you program using the *ACTIVE to ACTIVE Timing Register* on page 3-16.

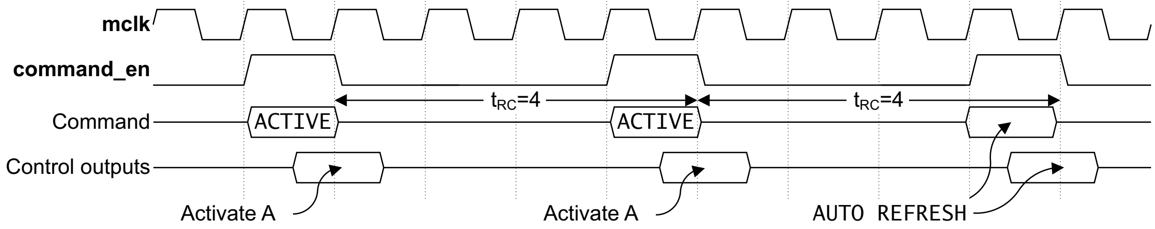


Figure 2-17 Same bank ACTIVE to ACTIVE, and ACTIVE to AUTO REFRESH timing, t_{RC}

Figure 2-18 shows the ACTIVE to ACTIVE command timing to different memory banks, that you program using the *ACTIVE to ACTIVE Different Bank Timing Register* on page 3-19.

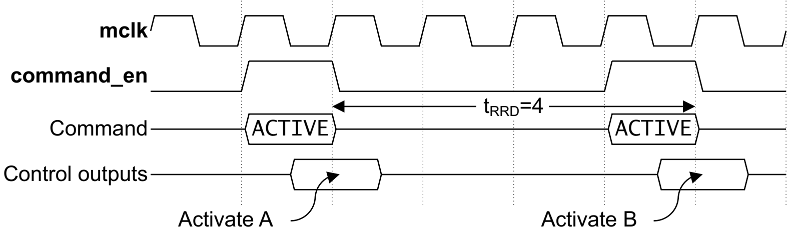


Figure 2-18 Different bank ACTIVE to ACTIVE command timing, t_{RRD}

Figure 2-19 shows the PRECHARGE to command, and AUTO REFRESH timing, that you program using the *PRECHARGE to Command Timing Register* on page 3-18 and *AUTO REFRESH to Command Timing Register* on page 3-18.

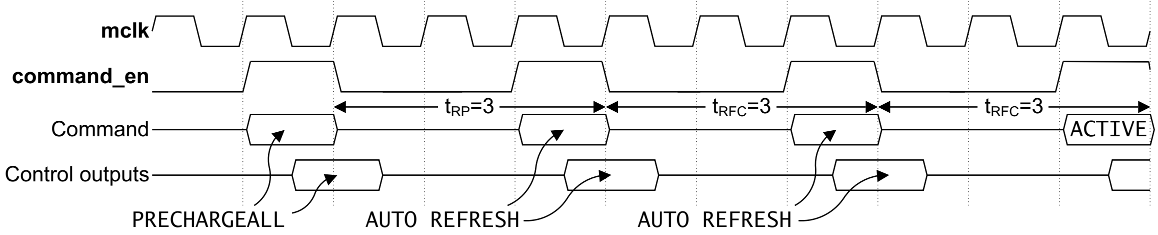


Figure 2-19 PRECHARGE to command and AUTO REFRESH to command timing, t_{RP} and t_{RFC}

Figure 2-20 on page 2-23 shows ACTIVE to PRECHARGE, and PRECHARGE to PRECHARGE timing, that you program using the *ACTIVE to PRECHARGE Timing Register* on page 3-16 and *PRECHARGE to Command Timing Register* on page 3-18.

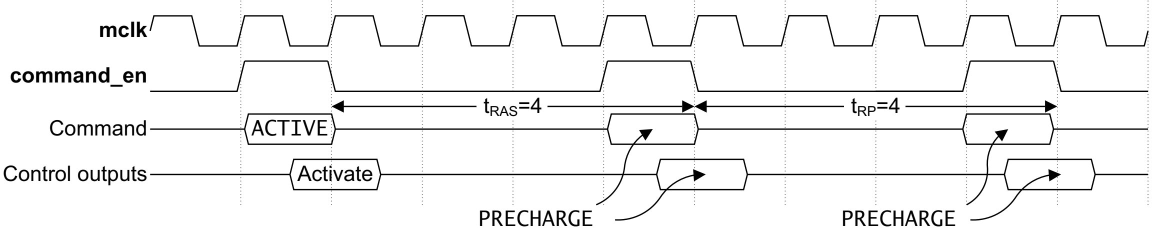


Figure 2-20 `ACTIVE` **to** `PRECHARGE`**, and** `PRECHARGE` **to** `PRECHARGE` **timing,** t_{RAS} **and** t_{RP}

Figure 2-21 shows `MODEREG` to command timing, that you program using the *MODEREG to Command Timing Register* on page 3-15.

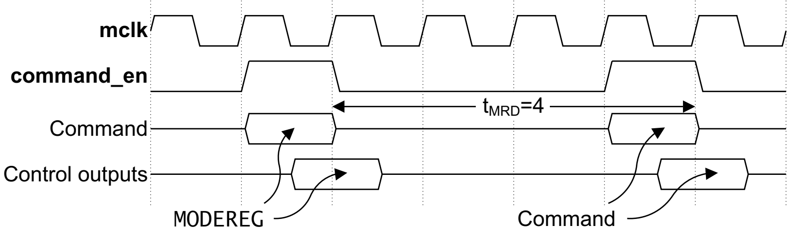


Figure 2-21 `MODEREG` **to command timing,** t_{MRD}

Figure 2-22 shows self-refresh entry and exit timing, that you program using the *Self-refresh to Command Timing Register* on page 3-22 and *Exit Self-refresh Timing Register* on page 3-21.

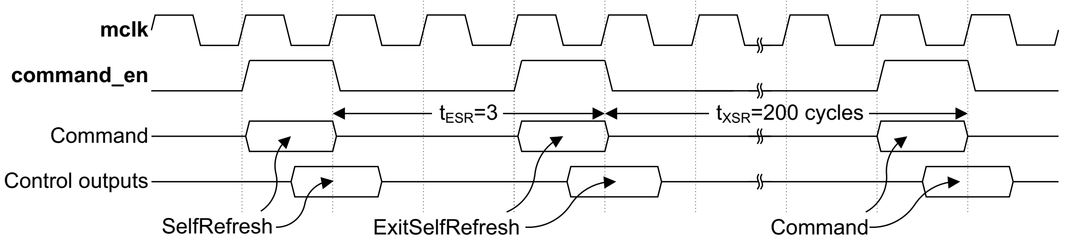


Figure 2-22 Self-refresh entry and exit timing, t_{ESR} **and** t_{XSR}

Figure 2-23 shows power-down entry and exit timing, that you program using the *Memory Configuration 3 Register* on page 3-24 and *Exit Power-down Timing Register* on page 3-20.

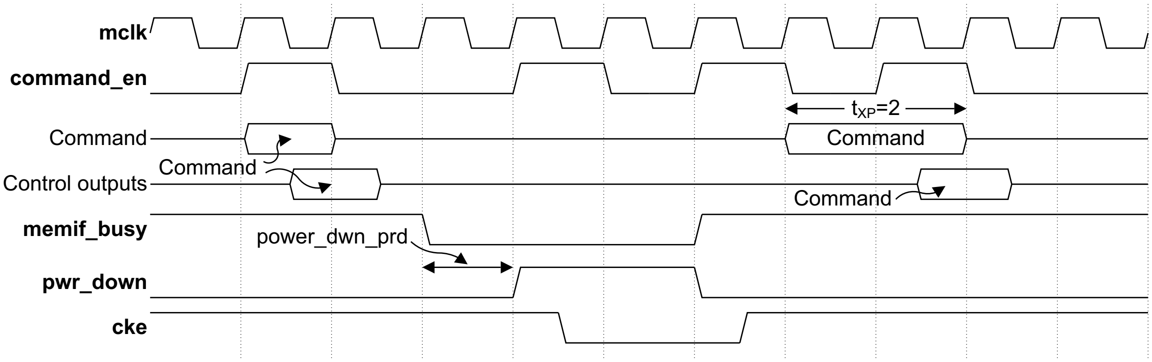


Figure 2-23 Power down entry and exit timing, t_{XP}

The power_dwn_prd count is timed from the memory interface becoming idle, that is, after a command delay has timed out or the read data FIFO is emptied. **cke** is asserted when the command FIFO is not empty.

Figure 2-24 shows the turnaround time, t_{WTR}, for the memory interface to output a `Write` command followed immediately by a `Read` command. Program this value using the *Write to Read Timing Register* on page 3-20.

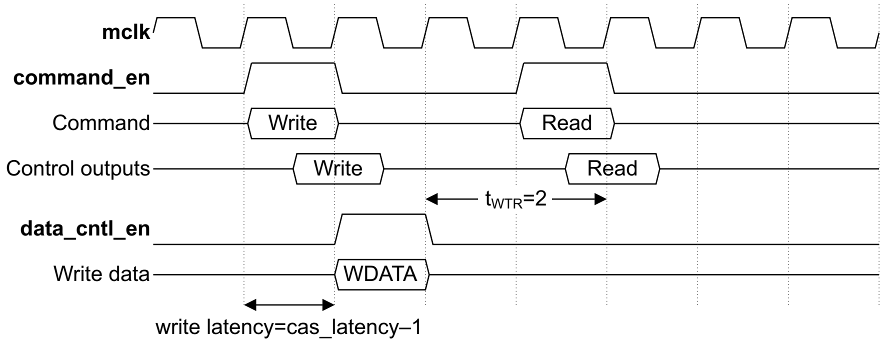


Figure 2-24 Data output timing, t_{WTR}

Figure 2-25 shows the relationship between the memory interface outputting a `Write` command and the write data, when CAS latency is set to 3, resulting in a write latency of 2. It also highlights the t_{WR} minimum time between a `Write` and a `PRECHARGE` command, that you program using the *Write to PRECHARGE Timing Register* on page 3-19.

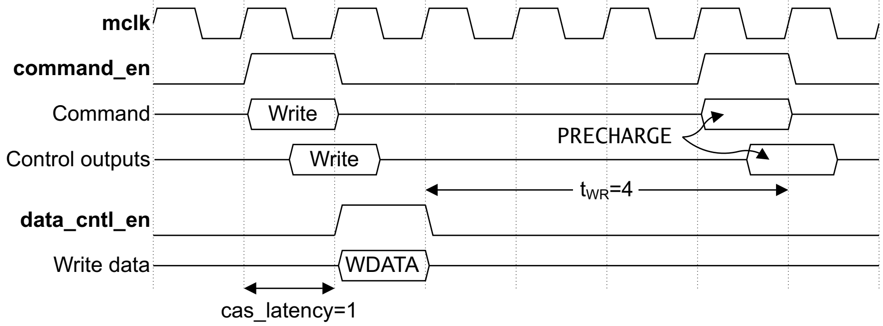


Figure 2-25 Data output timing, write latency = 2

Figure 2-26 on page 2-25 shows the timing relationship between a `Read` command being output from the memory interface and the read data being returned to the memory interface from the pad interface. Program this timing using the *CAS Latency Register* on page 3-14.

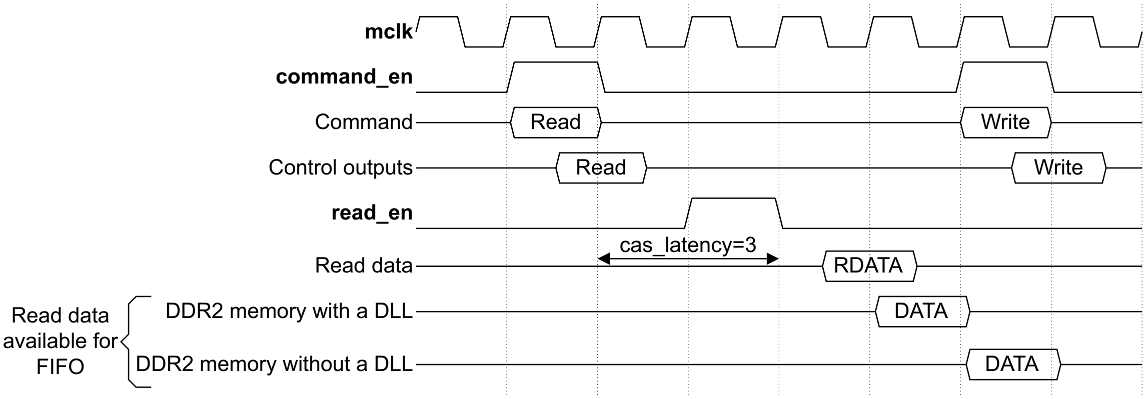


Figure 2-26 Data input timing

2.2.11 Pad interface

You can configure the DDR2 DMC can be configured to contain one of the following pad interfaces:

- *Legacy pad interface*
- *DFI pad interface* on page 2-26.

Legacy pad interface

The legacy pad interface is a replaceable block designed to operate with DDR2 SDRAM memory. It provides a register for each external signal.

You can replace or modify the legacy pad interface, if required, for additional optimization for a particular memory type or target library, or to use a hard macro.

If the legacy pad interface is modified or replaced, it is important that the relative timings of the control output signals enabled by **command_en** and **data_cntl_en** signals are maintained to ensure the timings carried out in the memory interface block are still correct at the external memory bus interface. The **read_en** signal is always asserted one **mclk** period before the expected read data. Therefore, the timing of **read_en** changes as cas_latency is changed using the APB interface.

When the controller issues a `Read` command, after a delay of several **mclk** cycles it registers the data into the read data FIFO, see Figure 2-26. This delay is dependent on the delays in the legacy pad interface and the value programmed in the cas_latency Register.

Legacy pad interface to external memory devices

The legacy pad interface block registers the relevant command signals with clocks that enable the external memory device timing to be met.

It is expected that a *Delay-Locked Loop* (DLL) is required to delay the **dqs_in_<n>**, **dqs_in_n_<n>**, **dqs_in_ecc** and **dqs_in_n_ecc** signals coming back from the memories with respect to the **dq_in** data bus. The standard delay for the **dqs_in_<n>**, **dqs_in_n_<n>**, **dqs_in_ecc** and **dqs_in_n_ecc** signals is a quarter clock period of **mclk**. A DLL is not included in the DDR2 DMC. The **asetbok** signal enables DLLs to update at safe points in time. See *asetbok signal* on page A-4.

DFI pad interface

If the DDR2 DMC is configured to support DFI then it implements a DFI pad interface that complies with Version 2.0 of the *DDR PHY Interface (DFI) Specification*.

2.2.12 Initialization

Before you can use the DDR2 DMC operationally to access external memory, you must:

- program the DDR2 DMC configuration and timing registers
- initialize the external memory devices by programming the direct_cmd Register.

You might not have to configure all of the DDR2 DMC registers because some might power-up to the correct value.

Note

A deadlock situation might occur if an AXI master accesses the AXI slave interface before a master has programmed the DDR2 DMC using the APB interface. Before the controller has been initialized, if a master can access the AXI slave interface but cannot access the APB interface, then that master must be held off until another master programs the controller.

The following sections provide examples of:

- *Controller initialization example*
- *DDR2 device initialization* on page 2-27.

Controller initialization example

Table 2-3 shows an example initialization programming sequence for the controller.

Table 2-3 Controller initialization example

Register	Write data	Description
cas_latency	`0x00000006`	Set CAS latency to 3
t_mrd	`0x00000002`	Set t_{MRD} to 2
t_ras	`0x00000008`	Set t_{RAS} to 8
t_rc	`0x0000000B`	Set t_{RC} to 11
t_rcd	`0x00000103`	Set t_{RCD} to 3 and schedule_rcd to 1
t_rfc	`0x00001315`	Set t_{RFC} to 21 and schedule_rfc to 19
t_rp	`0x00000103`	Set t_{RP} to 3 and schedule_rp to 1
t_rrd	`0x00000002`	Set t_{RRD} to 2
t_wr	`0x00000003`	Set t_{WR} to 3
t_wtr	`0x00000002`	Set t_{WTR} to 2
t_xp	`0x00000002`	Set t_{XP} to 2
t_xsr	`0x000000C8`	Set t_{XSR} to 200
t_esr	`0x00000003`	Set t_{ESR} to 3
t_rddata_en	`0x00000003`	Set t_{rddata_en} to 3 if the DDR2 DMC is configured with a DFI pad interface

Table 2-3 Controller initialization example (continued)

Register	Write data	Description
memory_cfg	`0x0001A411`	Sets the following memory configuration: • 9 column bits, 13 row bits • power-down period of 0 • disable auto power-down • disable dynamic clock stopping • memory burst size of 4 • use **arid[5:2]** bits for QoS
refresh_prd	`0x00000618`	Set an auto-refresh period of 3.9µs, that is, every 1560 (or `0x618`) **mclk** periods when **mclk** frequency is 400MHz
memory_cfg2	`0x00000001`	Sets the following memory configuration: • **aclk** and **mclk** are synchronous • **dqm[MEMORY_BYTES–1:0]** are LOW at reset • **cke** is LOW at reset • two bits for the bank address • 16-bit data bus
chip_cfg0	`0x000000FF`	Sets address for chip 0 to be `0x00XXXXXX`, *Row Bank Column* (RBC) configuration
chip_cfg1	`0x000022FF`	Sets address for chip 1 to be `0x22XXXXXX`, RBC configuration
chip_cfg2	`0x000055FF`	Sets address for chip 2 to be `0x55XXXXXX`, RBC configuration
chip_cfg3	`0x00007FFF`	Sets address for chip 3 to be `0x7FXXXXXX`, RBC configuration
Wait 200µs to enable the memory devices to stabilize		
direct_cmd	-	Sequence of writes to initialize the memory devices, see Table 2-4
memc_cmd	`0x00000000`	Change DDR2 DMC to the Ready state
memc_status	-	Poll the register until the memc_status field returns `0b01`, signifying that the controller is ready to accept AXI accesses to the memory devices

DDR2 device initialization

Table 2-4 shows an example programming sequence for DDR2 device initialization. See the *JEDEC STANDARD DDR2 SDRAM Specification* for more information.

Table 2-4 DDR2 device initialization

Register	Write data	Description
direct_cmd	`0x000C0000`	`NOP` command to chip 0.
Wait 400ns		
direct_cmd	`0x00000000`	`PRECHARGEALL` command to chip 0.
direct_cmd	`0x000A0000`	`MODEREG` command to extended mode register 2, with low address bits = `0x0`.
direct_cmd	`0x000B0000`	`MODEREG` command to extended mode register 3, with low address bits = `0x0`.
direct_cmd	`0x00090000`	`MODEREG` command to extended mode register 1, with low address bits = `0x0`, to enable the DLL.
direct_cmd	`0x00080132`	`MODEREG` command to mode register, with low address bits = `0x0132`. This resets the DLL, burst length = 4, sequential bursts, CAS latency = 3.

Table 2-4 DDR2 device initialization (continued)

Register	Write data	Description
direct_cmd	`0x00000000`	`PRECHARGEALL` command to chip 0.
direct_cmd	`0x00040000`	`AUTO REFRESH` command to chip 0.
direct_cmd	`0x00040000`	`AUTO REFRESH` command to chip 0.
direct_cmd	`0x00080032`	`MODEREG` command to mode register, with low address bits = `0x0032`. DLL is active, and sets burst length = 4, sequential bursts, CAS latency = 3.
Wait for 200 **mclk** periods		
direct_cmd	`0x00090380`	`MODEREG` command to extended mode register 1, with low address bits = `0x380`, to set OCD calibration default.
direct_cmd	`0x00090000`	`MODEREG` command to extended mode register 1, with low address bits = `0x0`, to exit OCD.

If a configured DDR2 DMC supports more than one memory chip then repeat the sequence in Table 2-4 on page 2-27 but update the chip_addr field, in the direct_cmd Register, to select each additional memory chip. See *Direct Command Register* on page 3-11.

2.2.13 Power-down support and usage model

The DDR2 DMC provides support for low-power operation in the following ways:

- By using the memory_cfg Register, the controller can automatically place a memory device into either the precharge power-down or active power-down states by deasserting **cke** when a memory device has been inactive for a number of clock cycles. This occurs with the controller in the Ready state. See *Memory Configuration Register* on page 3-12.
- By using the memc_cmd and memc_status Registers, the controller can place the memory device into self-refresh mode under software control. See *Memory Controller Status Register* on page 3-8 and *Memory Controller Command Register* on page 3-9.
- By using the AXI low-power interface, the controller can place the memory device into self-refresh mode under hardware control.

Additionally, the controller provides additional power savings through extensive use of clock gating.

It is possible to implement the controller with two power domains:

- APB and AXI, **aclk**
- memory, **mclk**. Figure 2-12 on page 2-8 shows these clock domains.

Table 2-5 shows the valid system states of the **aclk** FSM and an **mclk** FSM. It also shows the valid power, clock, and reset states in the **aclk** and **mclk** domains. Table 2-6 on page 2-30 shows the valid transitions, and the text following it explains how to traverse the system states.

Table 2-5 Valid system states for FSMs

State	Memory device		DDR2 DMC aclk FSM				DDR2 DMC mclk FSM			
	V_{DD}	State	V_{DD}	Clock	Reset	State	V_{DD}	Clock	Reset	State
1	0	Null	0	-	-	Null	0	-	-	Null
2	0	Null	>0	Running	No	POR	>0	Running	No	POR
3	0	Null	>0	Running	Yes	Reset	>0	Running	Yes	Reset
4	0	Null	>0	Running	No	Config	>0	Running	No	Pwr_up
5	>0	Accessible	>0	Running	No	Config	>0	Running	No	Pwr_up
6	>0	Accessible	>0	Running	No	Ready	>0	Running	No	Pwr_up
7	>0	Power-down	>0	Running	No	Ready	>0	Running	No	Pwr_down
8	>0	Self-refresh	>0	Running	No	Low_power	>0	Running	No	Pwr_sref
9	>0	Self-refresh	>0	Running	No	Low_power	>0	Stopped	No	Pwr_sref
10	>0	Self-refresh	>0	Stopped	No	Low_power	>0	Running	No	Pwr_sref
11	>0	Self-refresh	>0	Stopped	No	Low_power	>0	Stopped	No	Pwr_sref
12	>0	Self-refresh	0	-	-	Null	>0	Stopped	No	Pwr_sref
13	>0	Self-refresh	0	-	-	Null	>0	Running	No	Pwr_sref
14	>0	Self-refresh	>0	Running	No	POR	>0	Stopped	No	Pwr_sref
15	>0	Self-refresh	>0	Running	No	POR	>0	Running	No	Pwr_sref
16	>0	Self-refresh	>0	Running	Yes	Reset	>0	Stopped	No	Pwr_sref
17	>0	Self-refresh	>0	Running	Yes	Reset	>0	Running	No	Pwr_sref

The ranking of system power states, from highest power to lowest power, is as follows:

6, 7, 8, 10, 9, 11, 13, 12.

However, states 8-11 are similar and the recommendation is to use state 11 from this group if clock-stopping techniques are available. Similarly, states 12 & 13 are similar and the recommendation is to use state 12 from this pair. Table 2-6 shows a recommended set of power states.

Table 2-6 Recommended power states

System state	Power name
6	Running
7	Auto power-down
11	Shallow self-refresh
12	Deep self-refresh

Figure 2-27 shows these system states and arcs.

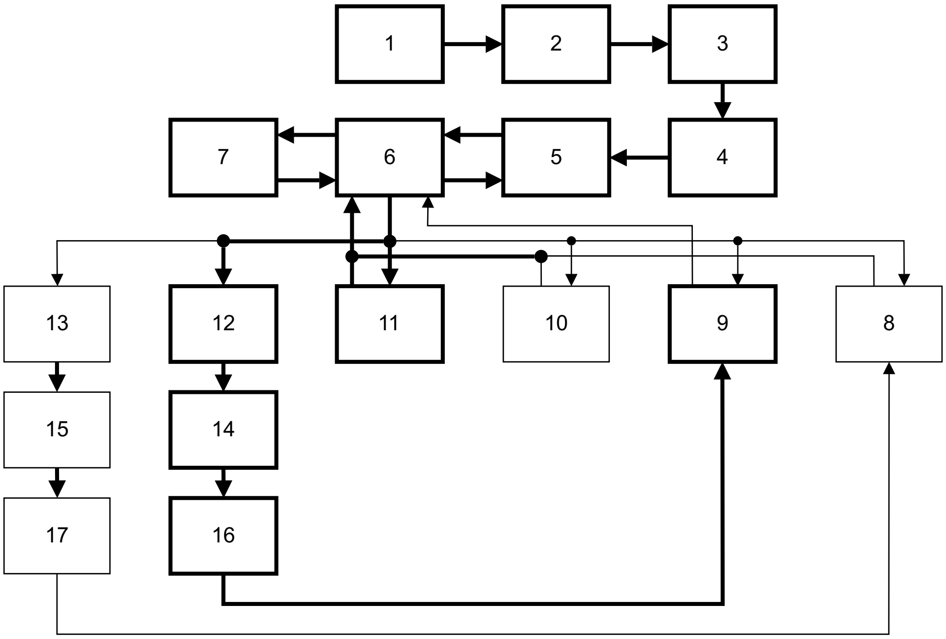


Figure 2-27 System state transitions

——— **Note** ———

States 1-5, 9, 14, and 16 are only used as transitional states.

————————

The state transitions are:

Arc 1 to 2 Apply power to all DDR2 DMC power domains, and ensure that **aclk** and **mclk** are running.

Arc 2 to 3 Assert reset in the **aclk** reset domain and the **mclk** reset domain.

Arc 3 to 4 Deassert reset in the **aclk** reset domain and the **mclk** reset domain.

Arc 4 to 5 Apply power to the memory device power domain.

Arc 5 to 6 You must:

1. Program the DDR2 DMC timing registers with the timing parameters for the DDR2 SDRAM. Figure 3-2 on page 3-3 shows the timing registers.
2. Write to the memory_cfg and refresh_prd Registers. See *Memory Configuration Register* on page 3-12 and *Refresh Period Register* on page 3-14.
3. Initialize the memory, using the direct_cmd Register, with the sequence of commands specified by the memory vendor. See *Direct Command Register* on page 3-11 and *DDR2 device initialization* on page 2-27.

   When you have sent these commands to the memory, you can write to the memc_cmd Register with the Go command. See *Memory Controller Command Register* on page 3-9.
4. Poll the memc_status Register until the memc_status field returns `0b01`, signifying that the controller is ready to accept AXI accesses to the memory devices.

Arc 6 to 5 If you want to reconfigure either the controller or memory devices, you must first write to the memc_cmd Register, with the Pause command, and poll the memc_status Register until the memc_status field returns `0b10`, Paused.

Then you can write to the memc_cmd Register with the Configure command, and poll the memc_status Register until the memc_status field returns `0b00`, Config. See *Memory Controller Status Register* on page 3-8 and *Memory Controller Command Register* on page 3-9.

Arc 6 to 7 If auto_power_down is set in the memory_cfg Register then this arc is automatically taken when the SDRAM has been idle for power_dwn_prd **mclk** cycles. See *Memory Configuration Register* on page 3-12.

Arc 7 to 6 When an SDRAM access command has been received in the **mclk** domain, this arc is taken.

Arc 6 to 8 You can take this arc under either hardware or software control:

- To take this arc under software control:
  1. Issue the Pause command using the memc_cmd Register.
  2. Poll the memc_status Register for the Paused state.
  3. Issue the Sleep command using the memc_cmd Register.
- To take this arc under hardware control, use the AXI low-power interface to request the Low_power state.

Arc 6 to 9 The same as arc 6 to 8 but also stops the **mclk** domain clock.

Arc 6 to 10 The same as arc 6 to 8 but also stops the **aclk** domain clock.

Arc 6 to 11 The same as arc 6 to 8 but also stop the **mclk** and the **aclk** domain clocks.

Arc 6 to 12 The same as arc 6 to 8 but also stops the **mclk** domain clock and removes power from the **aclk** power domain. This can only be done if the DDR2 DMC implementation has separate power domains for **aclk** and **mclk**.

Arc 6 to 13 The same as arc 6 to 8 but also removes power from the **aclk** power domain. This can only be done if the DDR2 DMC implementation has separate power domains for **aclk** and **mclk**.

Arc 8 to 6 You can take this arc under either hardware or software control:

- To take this arc under software control:
  1. Issue the Wakeup command using the memc_cmd Register.
  2. Poll the memc_status Register for the Paused state.
  3. Issue the Go command and poll for the Ready state.
- To take this arc under hardware control, use the AXI low-power interface to bring the controller out of the Low_power state.

Arc 9 to 6 The same as arc 8 to 6 but you must first start the **mclk** domain clock.

Arc 10 to 6 The same as arc 8 to 6 but you must first start the **aclk** domain clock.

Arc 11 to 6 The same as arc 8 to 6 but you must first start the **aclk** and **mclk** domain clocks.

Arc 12 to 14 Apply power to the **aclk** power domain.

Arc 14 to 16 Assert reset to the **aclk** reset domain.

Arc 16 to 9 Deassert reset to the **aclk** reset domain.

Arc 13 to 15 Apply power to the **aclk** power domain.

Arc 15 to 17 Assert reset to the **aclk** reset domain.

Arc 17 to 8 Deassert reset to the **aclk** reset domain.

Chapter 3
Programmers Model

This chapter describes the CoreLink DDR2 DMC registers and provides information about programming the controller. It contains the following sections:

3.1 About the programmers model

The following information applies to the DDR2 DMC registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these location can result in Unpredictable behavior.
- Unless otherwise stated in the accompanying text:
  - — do not modify undefined register bits
  - — ignore undefined register bits on reads
  - — all register bits are reset to a logic 0 by a system or power-on reset.
- Access type in *Register summary* on page 3-6 is described as follows:

  RW Read and write.

  RO Read only.

  WO Write only.

3.1.1 Register map

The register map of the DDR2 DMC spans a 4KB region, see Figure 3-1.

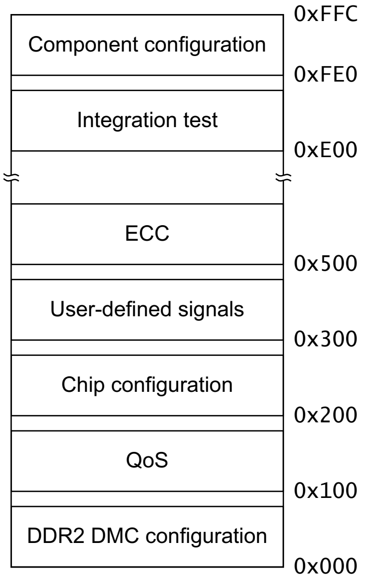


Figure 3-1 Register map

In Figure 3-1 the register map consists of the following main blocks:

- *DDR2 DMC configuration* on page 3-3
- *QoS registers* on page 3-3
- *Chip configuration* on page 3-4
- *User registers* on page 3-4
- *ECC registers* on page 3-4
- *Integration test* on page 3-4
- *Component configuration* on page 3-5.

DDR2 DMC configuration

Figure 3-2 shows the DDR2 DMC configuration register map.

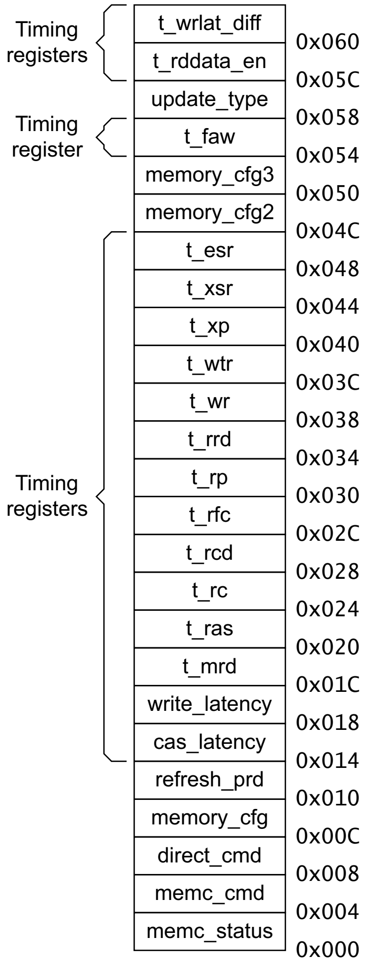


Figure 3-2 Configuration register map

QoS registers

Figure 3-3 shows the QoS register map.

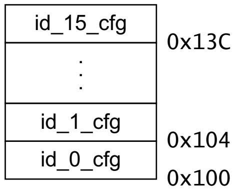


Figure 3-3 QoS register map

Chip configuration

Figure 3-4 shows the chip configuration register map.

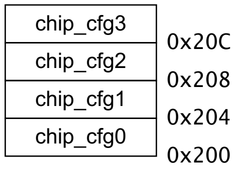


Figure 3-4 Chip configuration register map

User registers

Figure 3-5 shows the memory map for the Feature Control Register and the following user signals:

- **user_config1[ ]**
- **user_config0[ ]**
- **user_status[ ]**.

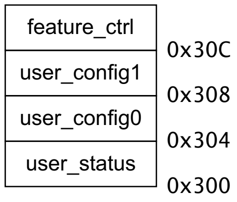


Figure 3-5 User register map

ECC registers

Figure 3-6 shows the *Error Correction Code* (ECC) memory map.

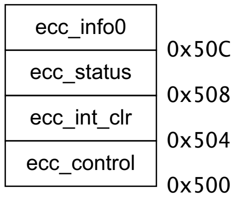


Figure 3-6 ECC configuration memory map

Integration test

Use these registers to verify correct integration of the DDR2 DMC within a system, by enabling non-AMBA signals to be set and read.

Component configuration

Figure 3-7 shows the CoreLink identification register map.

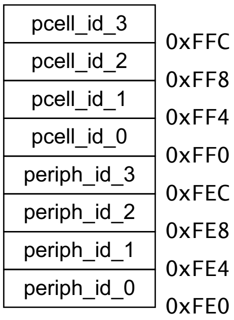


Figure 3-7 Component configuration register map

3.2 Register summary

Table 3-1 shows the DDR2 DMC registers in base offset order.

Table 3-1 DDR2 DMC register summary

Offset	Name	Type	Reset	Description
`0x000`	memc_status	RO	-[a]	*Memory Controller Status Register* on page 3-8
`0x004`	memc_cmd	WO	-	*Memory Controller Command Register* on page 3-9
`0x008`	direct_cmd	WO	-	*Direct Command Register* on page 3-11
`0x00C`	memory_cfg	RW	`0x00010021`	*Memory Configuration Register* on page 3-12
`0x010`	refresh_prd	RW	`0x000001E7`	*Refresh Period Register* on page 3-14
`0x014`	cas_latency	RW	`0x0000000A`	*CAS Latency Register* on page 3-14
`0x018`	write_latency	RO	`0x00000004`	*Write Latency Register* on page 3-15
`0x01C`	t_mrd	RW	`0x00000002`	*MODEREG to Command Timing Register* on page 3-15
`0x020`	t_ras	RW	`0x0000000E`	*ACTIVE to PRECHARGE Timing Register* on page 3-16
`0x024`	t_rc	RW	`0x00000012`	*ACTIVE to ACTIVE Timing Register* on page 3-16
`0x028`	t_rcd	RW	`0x00000205`	*ACTIVE to Read or Write Timing Register* on page 3-17
`0x02C`	t_rfc	RW	`0x00002023`	*AUTO REFRESH to Command Timing Register* on page 3-18
`0x030`	t_rp	RW	`0x00000205`	*PRECHARGE to Command Timing Register* on page 3-18
`0x034`	t_rrd	RW	`0x00000004`	*ACTIVE to ACTIVE Different Bank Timing Register* on page 3-19
`0x038`	t_wr	RW	`0x00000005`	*Write to PRECHARGE Timing Register* on page 3-19
`0x03C`	t_wtr	RW	`0x00000004`	*Write to Read Timing Register* on page 3-20
`0x040`	t_xp	RW	`0x00000002`	*Exit Power-down Timing Register* on page 3-20
`0x044`	t_xsr	RW	`0x00000027`	*Exit Self-refresh Timing Register* on page 3-21
`0x048`	t_esr	RW	`0x00000014`	*Self-refresh to Command Timing Register* on page 3-22
`0x04C`	memory_cfg2	RW	-[a]	*Memory Configuration 2 Register* on page 3-22
`0x050`	memory_cfg3	RW	`0x00000007`	*Memory Configuration 3 Register* on page 3-24
`0x054`	t_faw	RW	`0x00001114`	*Four Activate Window Timing Register* on page 3-24
`0x058`	update_type[b]	RW	`0x00000000`	*Update Type Register* on page 3-25
`0x05C`	t_rddata_en[b]	RW	`0x00000000`	*Read Data Enable Timing Register* on page 3-26
`0x060`	t_wrlat_diff[b]	RW	`0x00000000`	*Write Data Enable Timing Register* on page 3-27
`0x064-0x0FC`	-	-	-	Reserved, read undefined, write as zero
`0x100-0x13C`	id_<n>_cfg	RW	`0x00000000`	*QoS Configuration Registers* on page 3-27
`0x140-0x1FC`	-	-	-	Reserved, read undefined, write as zero

Table 3-1 DDR2 DMC register summary (continued)

Offset	Name	Type	Reset	Description
`0x200` `0x204`[c] `0x208`[c] `0x20C`[c]	chip_cfg0 chip_cfg1 chip_cfg2 chip_cfg3	RW	`0x0000FF00`	*Chip Configuration Registers* on page 3-28
`0x210-0x2FC`	-	-	-	Reserved, read undefined, write as zero
`0x300`	user_status	RO	-	*User Status Register* on page 3-29
`0x304`	user_config0	WO	-	*User Config 0 Register* on page 3-30
`0x308`	user_config1	WO	-	*User Config 1 Register* on page 3-30
`0x30C`	feature_ctrl	RW	`0x00000000`	*Feature Control Register* on page 3-31
`0x310-0x4FC`	-	-	-	Reserved, read undefined, write as zero
`0x500`	ecc_control[d]	RW	`0x00000000`	*ECC Control Register* on page 3-32
`0x504`	ecc_int_clr[d]	WO	-	*ECC Interrupt Clear Register* on page 3-32
`0x508`	ecc_status[d]	RO	`0x00000000`	*ECC Status Register* on page 3-33
`0x50C`	ecc_info0[d]	RO	`0x00000000`	*ECC Information Register* on page 3-34
`0x510-0xDFC`	-	-	-	Reserved, read undefined, write as zero

`0xE00`
`0xE04`
`0xE08`
	int_cfg
	int_inputs
	int_outputs
		See Chapter 4 *Programmers Model for Test* for more information about these registers		

`0xE0C-0xFDC`	-	-	-	Reserved, read undefined, write as zero
`0xFE0-0xFEC`	periph_id_n	RO	`0x00_41341`[e]	*Peripheral Identification Registers* on page 3-34
`0xFF0-0xFFC`	pcell_id_n	RO	`0xB105F00D`	*CoreLink Identification Registers* on page 3-36

[a] Dependent on configuration.

[b] This register is only present when the DDR2 DMC is configured to implement a *DDR PHY Interface* (DFI), otherwise reads are undefined, write as zero.

[c] The presence of this register depends on the number of chip selects that a configured controller supports. If a controller does not implement the register then reads are undefined, write as zero.

[d] This register is only present when the DDR2 DMC is configured to support ECC, otherwise reads are undefined, write as zero.

[e] Dependent on the revision of the DDR2 DMC, see *Peripheral Identification Register 2* on page 3-36.

3.3 Register descriptions

This section describes the DDR2 DMC registers.

3.3.1 Memory Controller Status Register

The memc_status Register characteristics are:

Purpose	Provides information about the configuration and current state of the DDR2 DMC.
Usage constraints	Not accessible in the Reset or *Power On Reset* (POR) state.
Configurations	Available in all configurations of the DDR2 DMC.
Attributes	See the register summary in Table 3-1 on page 3-6.

Figure 3-8 shows the memc_status Register bit assignments.

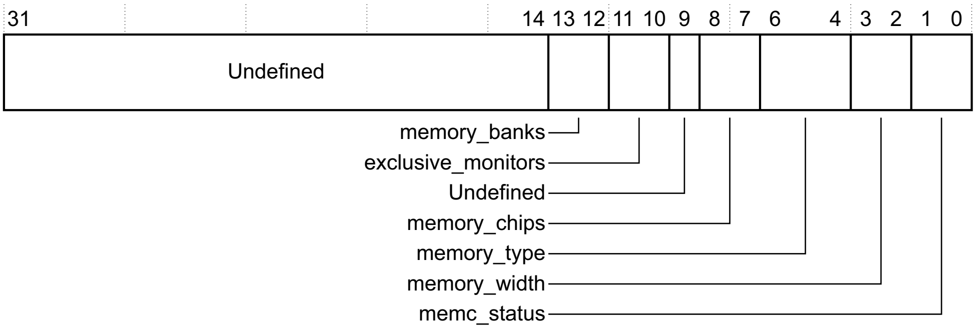


Figure 3-8 memc_status Register bit assignments

Table 3-2 shows the memc_status Register bit assignments.

Table 3-2 memc_status Register bit assignments

Bits	Name	Function
[31:14]	-	Read undefined.
[13:12]	memory_banks	Returns the maximum number of banks per memory chip: `0b00` = 4 banks `0b01` = 2 banks, not supported `0b10` = 1 banks, not supported `0b11` = 8 banks. This field is configured by the Memory Banks per Chip[a] option.
[11:10]	exclusive_monitors	Returns the number of exclusive access monitor resources implemented in the controller: `0b00` = 0 monitors `0b01` = 1 monitor `0b10` = 2 monitors `0b11` = 4 monitors. This field is configured by the Exclusive Monitors[a] option.
[9]	-	Undefined, read as zero.

Table 3-2 memc_status Register bit assignments (continued)

Bits	Name	Function
[8:7]	memory_chips	Returns the number of different chip selects that a configured controller supports: `0b00` = 1 chip `0b01` = 2 chips `0b10` = 3 chips `0b11` = 4 chips. This field is configured by the Memory Chips[a] option.
[6:4]	memory_type	Returns the type of SDRAM that the controller supports: `0b000`-`0b100` = reserved `0b101` = DDR2 SDRAM `0b110`- `0b111` = reserved.
[3:2]	memory_width	Returns the memory data bus width, MEMWIDTH, between the PHY and DDR2 SDRAM: `0b00` = 16-bit `0b01` = 32-bit `0b10` = 64-bit `0b11` = reserved. This field is configured by the Memory Bus Width[a] option.
[1:0]	memc_status	Returns the state of the memory controller: `0b00` = Config `0b01` = Ready `0b10` = Paused `0b11` = Low_power. **Note** In DFI implementations the status of the controller must match that of the memory when $t_{ctrl\_delay} > 1$. When software reads this field, it must wait t_{ctrl_delay} cycles before it can execute any action in the new state. For example, if entering Low_power to adjust the memory clock frequency, first the command is sent to the PHY before the controller updates its state. The system must wait t_{ctrl_delay} cycles before the self-refresh propagates to alter the clock.

[a] See the *CoreLink DDR2 Dynamic Memory Controller (DMC-341) Supplement to AMBA Designer (ADR-301) User Guide* for information about configuring this option.

3.3.2 Memory Controller Command Register

The memc_cmd Register characteristics are:

Purpose Controls the operating state of the DDR2 DMC.

Usage constraints Not accessible in the Reset or *Power On Reset* (POR) state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-9 on page 3-10 shows the memc_cmd Register bit assignments.

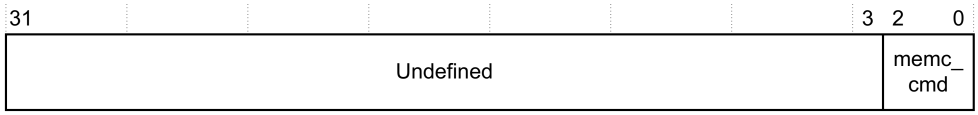


Figure 3-9 memc_cmd Register bit assignments

Table 3-3 shows the memc_cmd Register bit assignments.

Table 3-3 memc_cmd Register bit assignments

Bits	Name	Function
[31:3]	-	Undefined, write as zero.
[2:0]	memc_cmd	Use the following commands to change the state of the DDR2 DMC: `0b000` = Go `0b001` = Sleep `0b010` = Wakeup `0b011` = Pause `0b100` = Configure `0b111` = Active_Pause. If the controller receives a command to change state and a previous command to change state has not completed then it holds **pready** LOW until the new command can be carried out. See *aclk domain state diagram* on page 2-18 for more information about the state transitions.

Note

- Active_Pause command puts the DDR2 DMC into the Paused state without draining the arbiter queue. This enables you to move the controller to the Low_power state, to change configuration settings such as memory frequency or timing register values, without requiring co-ordination between masters in a multi-master system.
- If you use the Active_Pause command to put the controller in the Low_power state then you must not remove power from the controller because this results in data loss and violation of the AXI protocol.
- The controller does not issue refreshes when in the Config state. Therefore, ARM recommends that you make register updates with the controller in Low_power state because this ensures that the memory is put into self-refresh mode, rather than the Config state when the memory contains valid data.

3.3.3 Direct Command Register

The direct_cmd Register characteristics are:

Purpose Initializes and updates the external memory devices by sending the following commands:

- `NOP`
- `PRECHARGEALL`
- `AUTO REFRESH`
- `MODEREG`.

Usage constraints Only accessible in Config state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

The direct_cmd Register therefore enables any initialization sequence that an external memory device might require. The only timing information associated with the direct_cmd Register are the command delays that are programmed in the timing registers. Figure 3-2 on page 3-3 shows the timing registers. Therefore, if an initialization sequence requires additional delays between commands, they must be timed by the master driving the initialization sequence.

Figure 3-10 shows the direct_cmd Register bit assignments.

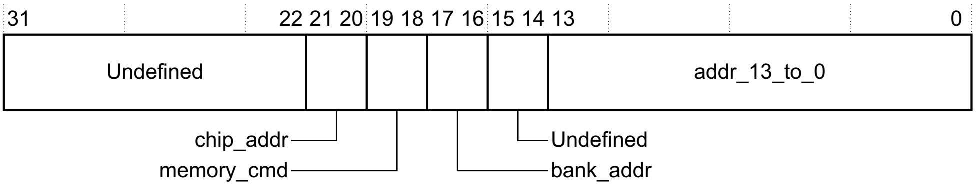


Figure 3-10 direct_cmd Register bit assignments

Table 3-4 shows the direct_cmd Register bit assignments.

Table 3-4 direct_cmd Register bit assignments

Bits	Name	Function
[31:22]	-	Undefined, write as zero.
[21:20]	chip_addr	Bits mapped to external memory chip selects, **cs_n[MEMORY_CHIPS–1:0]** or **dfi_cs_n[MEMORY_CHIPS–1:0]**.

[19:18]	memory_cmd	Selects the command required:
		`0b00` = `PRECHARGEALL`
		`0b01` = `AUTO REFRESH`
		`0b10` = `MODEREG` or Extended `MODEREG` access
		`0b11` = `NOP`.

[17:16]	bank_addr	Bits mapped to external memory bank address bits, **ba[1:0]** or **dfi_bank[1:0]**, when the controller issues a `MODEREG` command.
[15:14]	-	Undefined, write as zero.
[13:0]	addr_13_to_0	Bits mapped to external memory address bits, **add[13:0]** or **dfi_address[13:0]**, when the controller issues a `MODEREG` command.

3.3.4 Memory Configuration Register

The memory_cfg Register characteristics are:

Purpose Controls the operation of the DDR2 DMC.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-11 shows the memory_cfg Register bit assignments.

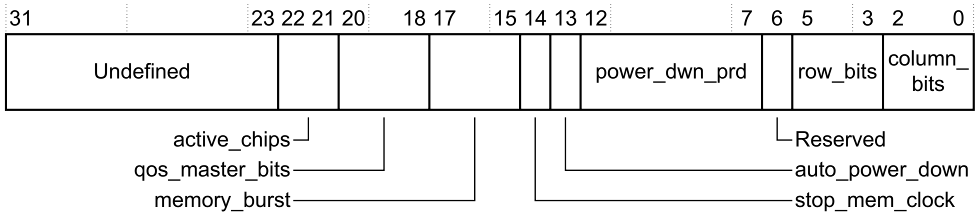


Figure 3-11 memory_cfg Register bit assignments

Table 3-5 shows the memory_cfg Register bit assignments.

Table 3-5 memory_cfg Register bit assignments

Bits	Name	Function
[31:23]	-	Read undefined, write as zero.
[22:21]	active_chips	Enables the DDR2 DMC to generate refresh commands for the following number of memory chips: `0b00` = 1 chip `0b01` = 2 chips `0b10` = 3 chips `0b11` = 4 chips. It is only possible to generate commands up to and including the number of chips in the configuration that the memc_status Register defines, see *Memory Controller Status Register* on page 3-8.
[20:18]	qos_master_bits	Controls which bits of the **arid** bus that the controller uses when it selects the QoS value for an AXI read transfer: `0b000` = **arid[3:0]** `0b001` = **arid[4:1]** `0b010` = **arid[5:2]** `0b011` = **arid[6:3]** `0b100` = **arid[7:4]** `0b101` = **arid[8:5]** `0b110` = **arid[9:6]** `0b111` = **arid[10:7]**. See *Quality of Service* on page 2-16 for more information.

Table 3-5 memory_cfg Register bit assignments (continued)

Bits	Name	Function
[17:15]	memory_burst	Controls how many data accesses that the controller performs to a memory device, for each `Read` or `Write` command: `0b010` = burst of 4 `0b011` = burst of 8 others = reserved. The chosen burst value must also be programmed into the mode register of the DDR2 SDRAM using the direct_cmd Register. See *Direct Command Register* on page 3-11.
[14]	stop_mem_clock	This enables the controller to stop the clock to the SDRAMs after the memory devices enter self-refresh mode. When set to 1, if the DDR2 DMC implements a: **Legacy pad interface** It dynamically stops the **clk_out[MEMORY_CHIPS–1:0]** signals. **DFI pad interface** It sets **dfi_dram_clk_disable[MEMORY_CHIPS–1:0]** HIGH.
[13]	auto_power_down	When this is set, the memory interface automatically places the DDR2 SDRAMs into power-down state by deasserting **cke**, or **dfi_cke**, when the command FIFO has been empty for power_dwn_prd memory clock cycles.
[12:7]	power_dwn_prd	Number of memory clock cycles for auto power-down of the DDR2 SDRAMs. You must only change this field when either: • auto_power_down bit is 0 • DDR2 DMC is in the Low_power state.
[6]	-	Reserved. Ignored for writes, read as zero.
[5:3]	row_bits	Encodes the number of bits of the AXI address that comprise the row address: `0b000` = reserved `0b001` = reserved `0b010` = 13 bits `0b011` = 14 bits `0b100` = 15 bits `0b101` = 16 bits `0b110` - `0b111` = reserved.
[2:0]	column_bits	Encodes the number of bits of the AXI address that comprise the column address: `0b000` = reserved. `0b001` = 9 bits. `0b010` = 10 bits. `0b011` = 11 bits. This means that A0-A9, and A11 are used for column address because A10 is a dedicated AP bit. `0b100` - `0b111` = reserved.

3.3.5 Refresh Period Register

The refresh_prd Register characteristics are:

Purpose Controls the memory refresh period in memory clock cycles.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-12 shows the refresh_prd Register bit assignments.

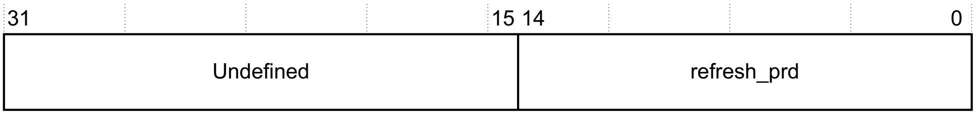


Figure 3-12 refresh_prd Register bit assignments

Table 3-6 shows the refresh_prd Register bit assignments.

Table 3-6 refresh_prd Register bit assignments

Bits	Name	Function
[31:15]	-	Read undefined, write as zero
[14:0]	refresh_prd	Memory refresh period in memory clock cycles. Supported values are 63-32767.

3.3.6 CAS Latency Register

The cas_latency Register characteristics are:

Purpose Controls the CAS latency time in memory clock cycles, see Figure 2-26 on page 2-25.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-13 shows the cas_latency Register bit assignments.

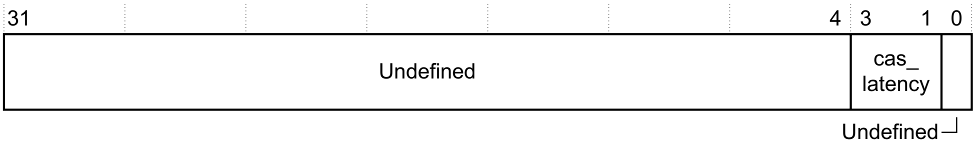


Figure 3-13 cas_latency Register bit assignments

Table 3-7 shows the cas_latency Register bit assignments.

Table 3-7 cas_latency Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:1]	cas_latency	CAS latency in **mclk** cycles. Supported values are 3-6.
[0]	-	Undefined, read and write as zero.

3.3.7 Write Latency Register

The write_latency Register characteristics are:

Purpose Returns the write latency in memory clock cycles.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-14 shows the write_latency Register bit assignments.

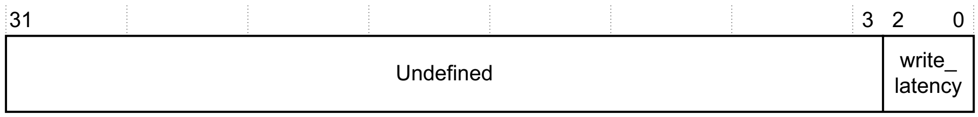


Figure 3-14 write_latency Register bit assignments

Table 3-8 shows the write_latency Register bit assignments.

Table 3-8 write_latency Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined.
[2:0]	write_latency	Returns the write latency in memory clock cycles. Supported values are 2-5. **Note** write_latency = cas_latency−1.

3.3.8 MODEREG to Command Timing Register

The t_mrd Register characteristics are:

Purpose Controls the MODEREG to command delay in memory clock cycles, see Figure 2-21 on page 2-23.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-15 on page 3-16 shows the t_mrd Register bit assignments.

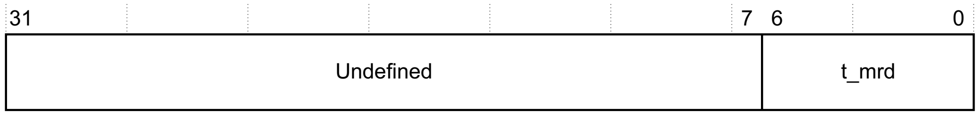


Figure 3-15 t_mrd Register bit assignments

Table 3-9 shows the t_mrd Register bit assignments.

Table 3-9 t_mrd Register bit assignments

Bits	Name	Function
[31:7]	-	Read undefined, write as zero
[6:0]	t_mrd	Sets t_{MRD}, the time delay, in **mclk** cycles, for the DDR2 DMC to issue a command after it issues a `MODEREG` command. Supported values are 1-127.

3.3.9 `ACTIVE` to `PRECHARGE` Timing Register

The t_ras Register characteristics are:

Purpose Controls the `ACTIVE` to `PRECHARGE` delay in memory clock cycles, see Figure 2-20 on page 2-23.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-16 shows the t_ras Register bit assignments.

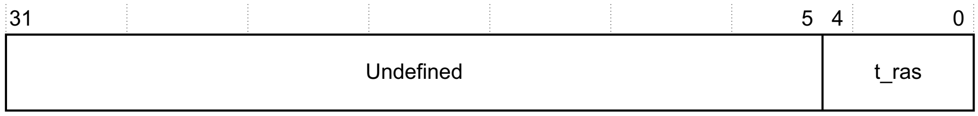


Figure 3-16 t_ras Register bit assignments

Table 3-10 shows the t_ras Register bit assignments.

Table 3-10 t_ras Register bit assignments

Bits	Name	Function
[31:5]	-	Read undefined, write as zero.
[4:0]	t_ras	Sets t_{RAS}, the `ACTIVE` to `PRECHARGE` delay in memory clock cycles. Supported values are 1-31.

3.3.10 `ACTIVE` to `ACTIVE` Timing Register

The t_rc Register characteristics are:

Purpose Controls the `ACTIVE` bank x to `ACTIVE` bank x delay in memory clock cycles, see Figure 2-17 on page 2-22.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-17 shows the t_rc Register bit assignments.

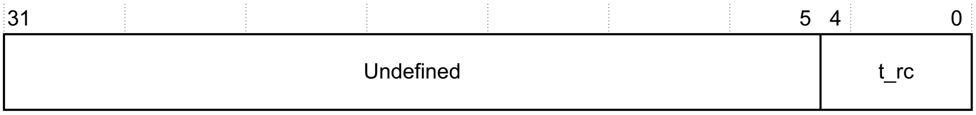


Figure 3-17 t_rc Register bit assignments

Table 3-11 shows the t_rc Register bit assignments.

Table 3-11 t_rc Register bit assignments

Bits	Name	Function
[31:5]	-	Read undefined, write as zero.
[4:0]	t_rc	Sets t_{RC}, the ACTIVE bank x to ACTIVE bank x delay in memory clock cycles. Supported values are 1-31.

3.3.11 ACTIVE to Read or Write Timing Register

The t_rcd Register characteristics are:

Purpose Controls the RAS to CAS minimum delay in memory clock cycles and controls the delay between an ACTIVE command and another memory command, other than ACTIVE, to the same bank, see Figure 2-15 on page 2-21.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-18 shows the t_rcd Register bit assignments.

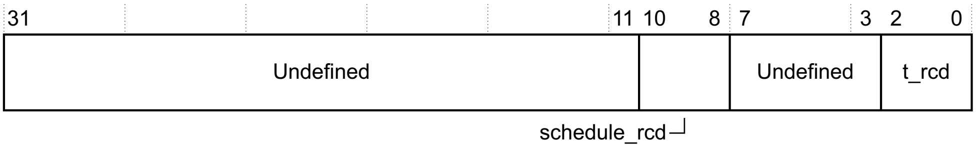


Figure 3-18 t_rcd Register bit assignments

Table 3-12 shows the t_rcd Register bit assignments.

Table 3-12 t_rcd Register bit assignments

Bits	Name	Function
[31:11]	-	Read undefined, write as zero.
[10:8]	schedule_rcd	Sets the RAS to CAS minimum delay in **aclk** clock cycles minus 3. It is used as a scheduler delay and values in the range 0-4 are supported.
[7:3]	-	Read undefined, write as zero.
[2:0]	t_rcd	Sets t_{RCD}, the RAS to CAS minimum delay in memory clock cycles. Supported values are 1-7.

3.3.12 AUTO REFRESH to Command Timing Register

The t_rfc Register characteristics are:

Purpose Controls the AUTO REFRESH to command delay in memory clock cycles, see Figure 2-19 on page 2-22.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-19 shows the t_rfc Register bit assignments.

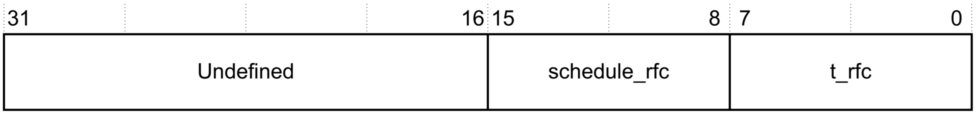


Figure 3-19 t_rfc Register bit assignments

Table 3-13 shows the t_rfc Register bit assignments.

Table 3-13 t_rfc Register bit assignments

Bits	Name	Function
[31:16]	-	Read undefined, write as zero.
[15:8]	schedule_rfc	Sets the AUTO REFRESH to command delay in **aclk** clock cycles minus 3. It is used as a scheduler delay and values in the range 0-255 are supported.
[7:0]	t_rfc	Sets t_{RFC}, the AUTO REFRESH to command delay in memory clock cycles. Supported values are 1-255.

3.3.13 PRECHARGE to Command Timing Register

The t_rp Register characteristics are:

Purpose Controls the PRECHARGE to RAS delay in memory clock cycles, see Figure 2-19 on page 2-22 and Figure 2-20 on page 2-23.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-20 shows the t_rp Register bit assignments.

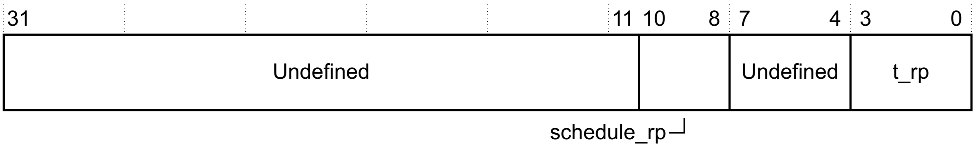


Figure 3-20 t_rp Register bit assignments

Table 3-14 shows the t_rp Register bit assignments.

Table 3-14 t_rp Register bit assignments

Bits	Name	Function
[31:11]	-	Read undefined, write as zero.
[10:8]	schedule_rp	Sets the PRECHARGE to RAS delay in **aclk** clock cycles minus 3. It is used as a scheduler delay and values in the range 0-4 are supported.
[7:4]	-	Read undefined, write as zero.
[3:0]	t_rp	Sets t_{RP}, the PRECHARGE to RAS delay in memory clock cycles. Supported values are 1-7.

3.3.14 ACTIVE to ACTIVE Different Bank Timing Register

The t_rrd Register characteristics are:

Purpose Controls the ACTIVE bank x to ACTIVE bank y delay in memory clock cycles, see Figure 2-18 on page 2-22.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-21 shows the t_rrd Register bit assignments.

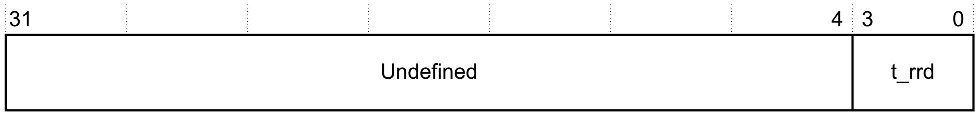


Figure 3-21 t_rrd Register bit assignments

Table 3-15 shows the t_rrd Register bit assignments.

Table 3-15 t_rrd Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:0]	t_rrd	Sets t_{RRD}, the ACTIVE bank x to ACTIVE bank y delay in memory clock cycles. Supported values are 1-15.

3.3.15 Write to PRECHARGE Timing Register

The t_wr Register characteristics are:

Purpose Controls the Write to PRECHARGE delay in memory clock cycles, see Figure 2-25 on page 2-24.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-22 on page 3-20 shows the t_wr Register bit assignments.

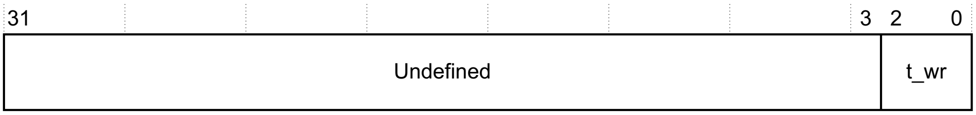


Figure 3-22 t_wr Register bit assignments

Table 3-16 shows the t_wr Register bit assignments.

Table 3-16 t_wr Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_wr	Sets t_{WR}, the `Write` to `PRECHARGE` delay in memory clock cycles. Supported values are 2-6.

3.3.16 `Write` to `Read` Timing Register

The t_wtr Register characteristics are:

Purpose Controls the `Write` to `Read` delay in memory clock cycles, see Figure 2-24 on page 2-24.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-23 shows the t_wtr Register bit assignments.

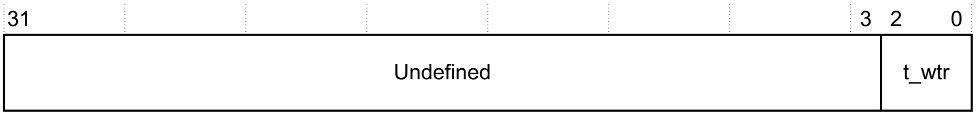


Figure 3-23 t_wtr Register bit assignments

Table 3-17 shows the t_wtr Register bit assignments.

Table 3-17 t_wtr Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_wtr	Sets t_{WTR}, the `Write` to `Read` command delay in memory clock cycles. Supported values are 1-7.

3.3.17 Exit Power-down Timing Register

The t_xp Register characteristics are:

Purpose Controls the exit power-down to command delay in memory clock cycles, see Figure 2-23 on page 2-23.

Usage constraints Only accessible in Config or Low_power state.

You must only write when either:

- auto_power_down bit is 0, see Table 3-5 on page 3-12
- DDR2 DMC is in the Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-24 shows the t_xp Register bit assignments.

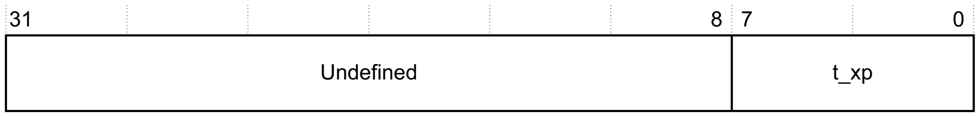


Figure 3-24 t_xp Register bit assignments

Table 3-18 shows the t_xp Register bit assignments.

Table 3-18 t_xp Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:0]	t_xp	Sets t_{XP}, the exit power-down to command delay in memory clock cycles. Supported values are 1-255.

3.3.18 Exit Self-refresh Timing Register

The t_xsr Register characteristics are:

Purpose Controls the exit self-refresh to command delay in memory clock cycles, see Figure 2-22 on page 2-23.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-25 shows the t_xsr Register bit assignments.

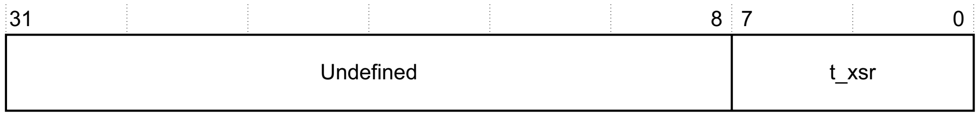


Figure 3-25 t_xsr Register bit assignments

Table 3-19 shows the t_xsr Register bit assignments.

Table 3-19 t_xsr Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:0]	t_xsr	Sets t_{XSR}, the exit self-refresh to command delay in memory clock cycles. Supported values are 1-255.

3.3.19 Self-refresh to Command Timing Register

The t_esr Register characteristics are:

Purpose Controls the self-refresh to command delay in memory clock cycles, see Figure 2-22 on page 2-23.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-26 shows the t_esr Register bit assignments.

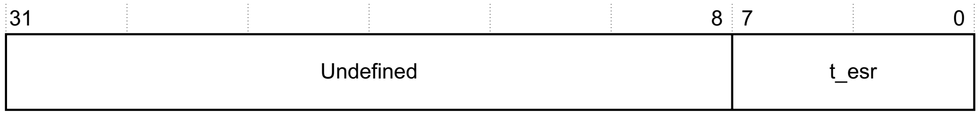


Figure 3-26 t_esr Register bit assignments

Table 3-20 shows the t_esr Register bit assignments.

Table 3-20 t_esr Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:0]	t_esr	Sets the self-refresh to command delay in memory clock cycles. Supported values are 1-255.

3.3.20 Memory Configuration 2 Register

The memory_cfg2 Register characteristics are:

Purpose Controls the operation of the DDR2 DMC. Enables you to override the configuration set by the tie-off signals, see *Tie-offs* on page A-3.

Usage constraints
- Only accessible in Config or Low_power state.
- Some bits are only available when a controller is configured to provide a legacy pad interface.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-27 shows the memory_cfg2 Register bit assignments.

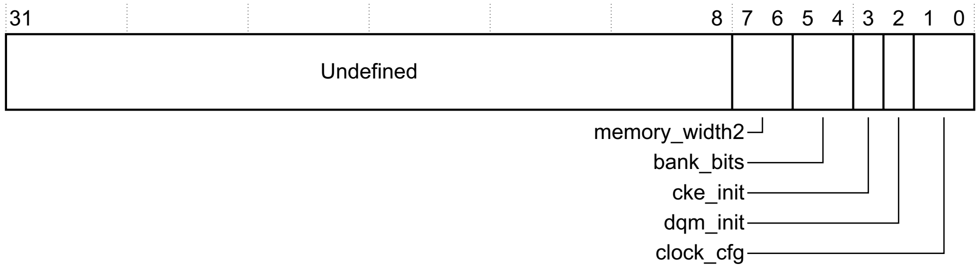


Figure 3-27 memory_cfg2 Register bit assignments

Table 3-21 shows the memory_cfg2 Register bit assignments.

Table 3-21 memory_cfg2 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:6]	memory_width2	Controls if the DDR2 DMC uses the entire configured memory data bus width or half of the data bus. Depending on the configured MEMWIDTH value, the following bit settings are possible: • `0b00` = memory device uses a 16-bit data bus. You can select this setting when MEMWIDTH=16 or MEMWIDTH=32. The controller uses the entire data bus when MEMWIDTH=16 and half of the data bus when MEMWIDTH=32. **Note** Do not use this setting if the controller uses an AXI data width of 128bits. See *Supported memory widths* on page 1-3. • `0b01` = memory device uses a 32-bit data bus.You can select this setting when MEMWIDTH=32 or MEMWIDTH=64. The controller uses the entire data bus when MEMWIDTH=32 and half of the data bus when MEMWIDTH=64. • `0b10` = memory device uses a 64-bit data bus.You can select this setting when MEMWIDTH=64. The controller uses the entire data bus. • `0b11` = reserved. You can determine the configured data bus width, MEMWIDTH, by accessing the memory_width field in the memc_status Register, see *Memory Controller Status Register* on page 3-8. The default value is set by the state of **memory_width[1:0]**, when **aresetn** goes HIGH.
[5:4]	bank_bits	Controls how many bits of the AXI address are allocated for addressing the banks in a memory device. The options are: `0b00` = 2 bits. Use this when a memory device contains a maximum of 4 banks. `0b01` = reserved. `0b10` = reserved. `0b11` = 3 bits. Use this when a memory device contains 8 banks and the controller configuration supports 8 banks. You can determine the configured number of supported memory banks by accessing the memory_banks field in the memc_status Register, see *Memory Controller Status Register* on page 3-8. The default value is set by the state of **bank_bits[1:0]**, when **aresetn** goes HIGH.
[3]	cke_init	If the DDR2 DMC contains a DFI pad interface then this bit is reserved. Otherwise, for a controller with a legacy pad interface, it sets the state of **cke** when **mresetn** is deasserted. The default value is set by the state of **cke_init**, when **aresetn** goes HIGH.
[2]	dqm_init	If the DDR2 DMC contains a DFI pad interface then this bit is reserved. Otherwise, for a controller with a legacy pad interface, it sets the state of the **dqm[MEMORY_BYTES–1:0]** outputs when **mresetn** is deasserted. The default value is set by the state of **dqm_init**, when **aresetn** goes HIGH.
[1:0]	clock_cfg	Encodes the clocking scheme: `0b00` = **aclk** and **mclk** are asynchronous `0b01` = **aclk** and **mclk** are synchronous `0b10` - `0b11` = reserved. The default value of bit [0] is set by the state of **sync**, when **aresetn** goes HIGH.

3.3.21 Memory Configuration 3 Register

The memory_cfg3 Register characteristics are:

Purpose Controls the refresh timeout setting.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-28 shows the memory_cfg3 Register bit assignments.

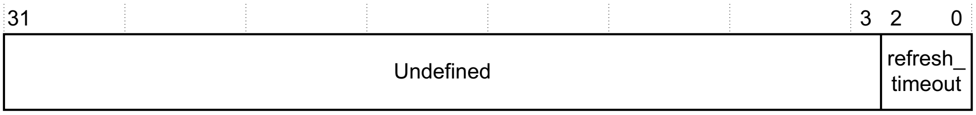


Figure 3-28 memory_cfg3 Register bit assignments

Table 3-22 shows the memory_cfg3 Register bit assignments.

Table 3-22 memory_cfg3 Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	refresh_timeout	Sets the number of acceptable outstanding refreshes for a memory device, before the DDR2 DMC generates a refresh timeout. If this occurs, the arbiter entry moves to the highest priority in the queue, see *Arbitration algorithm* on page 2-17. Supported values are 1-7.

3.3.22 Four Activate Window Timing Register

The t_faw Register characteristics are:

Purpose Controls the four bank activate time in memory clock cycles, see Figure 2-16 on page 2-22.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-29 shows the t_faw Register bit assignments.

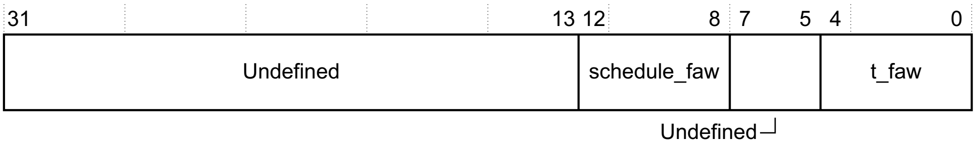


Figure 3-29 t_faw Register bit assignments

Table 3-23 shows the t_faw Register bit assignments.

Table 3-23 t_faw Register bit assignments

Bits	Name	Function
[31:13]	-	Read undefined, write as zero.
[12:8]	schedule_faw	t_faw in **aclk** clock cycles, minus 3. Used as a scheduler delay. Supported values are 0-31.
[7:5]	-	Read undefined, write as zero.
[4:0]	t_faw	Sets t_{FAW}, the four-bank activate period in memory clock cycles. Supported values are 1-31.

3.3.23 Update Type Register

The update_type Register characteristics are:

Purpose Controls how the DDR2 DMC responds when it receives any of the four possible update type requests from a PHY device.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available when a DDR2 DMC is configured to support a DFI pad interface.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-30 shows the update_type Register bit assignments.

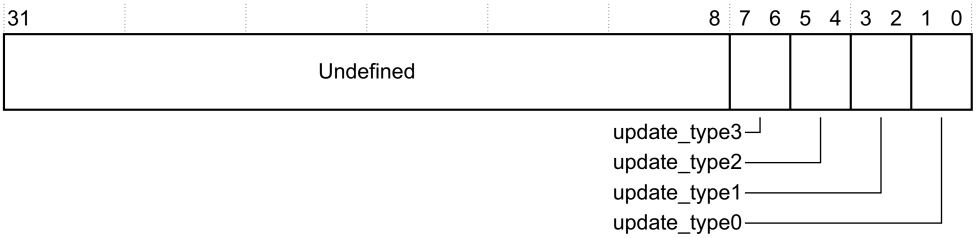


Figure 3-30 update_type Register bit assignments

Table 3-24 shows the update_type Register bit assignments.

Table 3-24 update_type Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:6]	update_type3	Controls how the DDR2 DMC responds to a DFI update request type 3, that is, when a PHY sets **dfi_phyupd_type[1:0]** to `0b11`: `0b00` = Put the memory devices in self-refresh mode then stall the DFI. `0b01` = Stall the DFI. `0b10` - `0b11` = Reserved.

Table 3-24 update_type Register bit assignments (continued)

Bits	Name	Function
[5:4]	update_type2	Controls how the DDR2 DMC responds to a DFI update request type 2, that is, when a PHY sets **dfi_phyupd_type[1:0]** to `0b10`: `0b00` = Put the memory devices in self-refresh mode then stall the DFI. `0b01` = Stall the DFI. `0b10-0b11` = Reserved.
[3:2]	update_type1	Controls how the DDR2 DMC responds to a DFI update request type 1, that is, when a PHY sets **dfi_phyupd_type[1:0]** to `0b01`: `0b00` = Put the memory devices in self-refresh mode then stall the DFI. `0b01` = Stall the DFI. `0b10- 0b11` = Reserved.
[1:0]	update_type0	Controls how the DDR2 DMC responds to a DFI update request type 0, that is, when a PHY sets **dfi_phyupd_type[1:0]** to `0b00`: `0b00` = Put the memory devices in self-refresh mode then stall the DFI. `0b01` = Stall the DFI. `0b10- 0b11` = Reserved.

3.3.24 Read Data Enable Timing Register

The t_rddata_en Register characteristics are:

Purpose Controls the t_{rddata_en} timing parameter for a DFI pad interface.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available when a DDR2 DMC is configured to support a DFI pad interface.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-31 shows the t_rddata_en Register bit assignments.

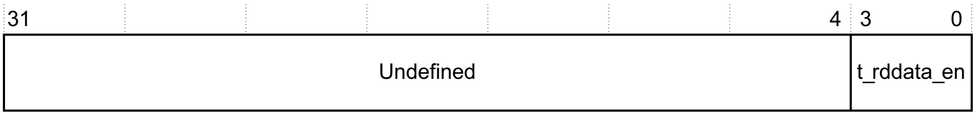


Figure 3-31 t_rddata_en Register bit assignments

Table 3-25 shows the t_rddata_en Register bit assignments.

Table 3-25 t_rddata_en Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:0]	t_rddata_en	After a DDR2 DMC issues a `Read` command from the DFI pad interface then this field sets the number of **mclk** cycles before **dfi_rddata_en[MEMORY_BYTES–1:0]** goes HIGH. The valid values for this field depend on the cas_latency setting and the limits are: **Minimum** cas_latency–2 **Maximum** cas_latency+5 For example, if cas_latency = 3 then set this field to any value between `0b0001` and `0b1000`. The cas_latency value is set using the *CAS Latency Register* on page 3-14.

3.3.25 Write Data Enable Timing Register

The t_wrlat_diff Register characteristics are:

Purpose Controls the t_{phy_wrlat} timing parameter for a DFI pad interface to be either:

- write_latency−1
- write_latency−2.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available when a DDR2 DMC is configured to support a DFI pad interface.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-32 shows the t_wrlat_diff Register bit assignments.

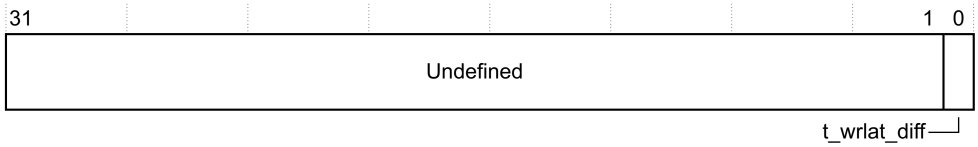


Figure 3-32 t_wrlat_diff Register bit assignments

Table 3-26 shows the t_wrlat_diff Register bit assignments.

Table 3-26 t_wrlat_diff Register bit assignments

Bits	Name	Function
[31:1]	-	Read undefined, write as zero.
[0]	t_wrlat_diff	Controls the timing relationship between t_{phy_wrlat} and write_latency to be either: 0 = t_{phy_wrlat} is write_latency−1. This is the default. 1 = t_{phy_wrlat} is write_latency−2. The write_latency value is set using the *Write Latency Register* on page 3-15.

3.3.26 QoS Configuration Registers

The id_<n>_cfg Register characteristics are:

Purpose Sets the parameters for QoS <n>. Where <n> is a value from 0 to 15 and is the result of applying the 4-bit QoS mask to the **arid[]** bus.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-33 on page 3-28 shows the id_<n>_cfg Register bit assignments.

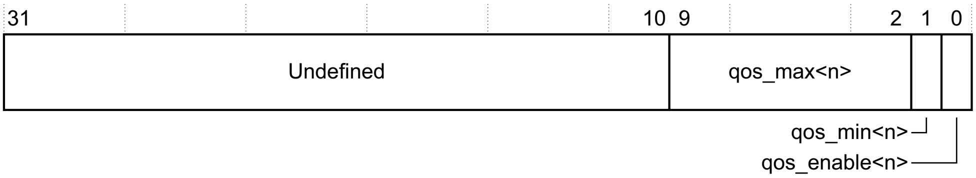


Figure 3-33 id_<n>_cfg Registers bit assignments

Table 3-27 shows the id_<n>_cfg Register bit assignments.

Table 3-27 id_<n>_cfg Registers bit assignments

Bits	Name	Function
[31:10]	-	Read undefined, write as zero.
[9:2]	qos_max<n>	Sets the maximum QoS value. Supported values are 0-255.
[1]	qos_min<n>	Enables the minimum QoS functionality: 0 = disabled, so the arbiter entry uses the qos_max<n> value 1 = enabled. the arbiter entry uses minimum latency.
[0]	qos_enable<n>	When set, the DDR2 DMC can apply QoS to a read transfer if the masking of the **arid[ ]** bits with the programmed QoS mask produces a value of <n>. See *Quality of Service* on page 2-16 for more information.

3.3.27 Chip Configuration Registers

The chip_cfg<n> Register characteristics are:

Purpose Each register sets the:

- address decode for chip select <n>
- bank, row, column organization of the memory device that connects to chip select <n>.

Usage constraints

- Only accessible in Config or Low_power state.
- The number of registers implemented is equal to the number of chip selects that a configured controller supports.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-34 shows the chip_cfg<n> Register bit assignments.

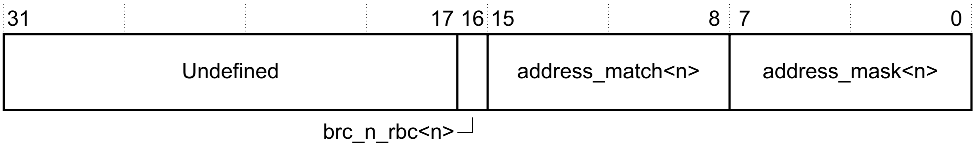


Figure 3-34 chip_cfg<n> Registers bit assignments

Table 3-28 shows the chip_cfg<n> Register bit assignments.

Table 3-28 chip_cfg<n> Registers bit assignments

Bits	Name	Function
[31:17]	-	Read undefined, write as zero.
[16]	brc_n_rbc<n>	Selects the memory organization as decoded from the AXI address: 0 = *Row, Bank, Column* (RBC) organization 1 = *Bank, Row, Column* (BRC) organization.
[15:8]	address_match<n>	The controller applies the address_mask<n> field to the AXI address bits, [31:24], that is **awaddr[31:24]** or **araddr[31:24]**. The controller compares the result against this field and if a match occurs then it selects memory device <n>. See *Formatting from AXI address channels* on page 2-11.
[7:0]	address_mask<n>	Controls which AXI address bits [31:24] the DDR2 DMC compares when it receives an AXI transfer: **Bit [*x*] = 0** DDR2 DMC excludes AXI address bit [24+*x*] from the comparison **Bit [*x*] = 1** DDR2 DMC includes AXI address bit [24+*x*] in the comparison.

Note

If a configured controller supports two or more memory devices then you must take care to ensure that for all AXI addresses, you program the various address_match and address_mask fields so that the controller can only assert a single memory chip select. Otherwise, Unpredictable behavior might occur.

3.3.28 User Status Register

The user_status Register characteristics are:

Purpose Provides the status of the **user_status** inputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-35 shows the user_status Register bit assignments.

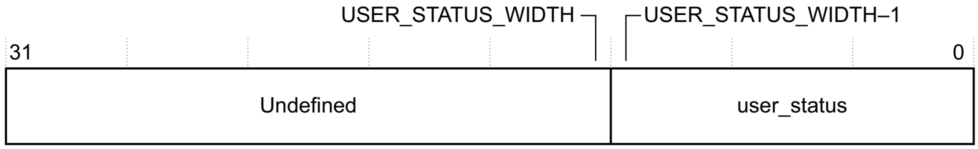


Figure 3-35 user_status Register bit assignments

Table 3-29 shows the user_status Register bit assignments.

Table 3-29 user_status Register bit assignments

Bits	Name	Function
[31:USER_STATUS_WIDTH[a]]	-	Read undefined
[USER_STATUS_WIDTH–1:0]	user_status	The state of the **user_status[USER_STATUS_WIDTH–1:0]** inputs

[a] If USER_STATUS_WIDTH is configured as 32 then none of the bits in this register are undefined.

3.3.29 User Config 0 Register

The user_config0 Register characteristics are:

Purpose Controls the state of the **user_config0** outputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-36 shows the user_config0 Register bit assignments.

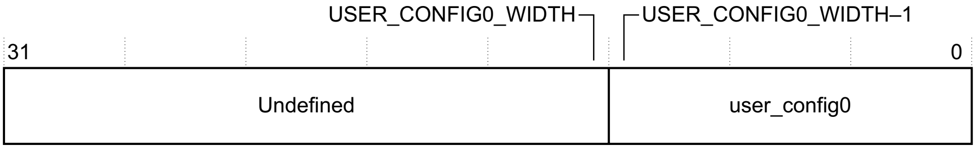


Figure 3-36 user_config0 Register bit assignments

Table 3-30 shows the user_config0 Register bit assignments.

Table 3-30 user_config0 Register bit assignments

Bits	Name	Function
[31:USER_CONFIG0_WIDTH[a]]	-	Undefined, write as zero
[USER_CONFIG0_WIDTH–1:0]	user_config0	Sets the state of the **user_config0[USER_CONFIG0_WIDTH–1:0]** outputs

[a] If USER_CONFIG0_WIDTH is configured as 32 then none of the bits in this register are undefined.

3.3.30 User Config 1 Register

The user_config1 Register characteristics are:

Purpose Controls the state of the **user_config1** outputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-37 shows the user_config1 Register bit assignments.

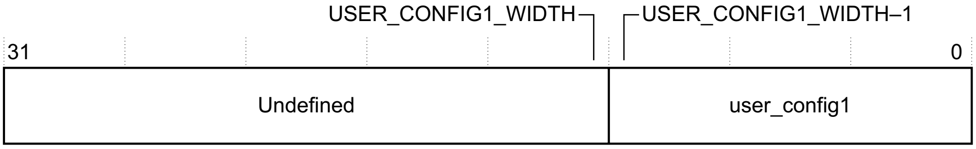


Figure 3-37 user_config1 Register bit assignments

Table 3-31 shows the user_config1 Register bit assignments.

Table 3-31 user_config1 Register bit assignments

Bits	Name	Function
[31:USER_CONFIG1_WIDTH[a]]	-	Undefined, write as zero
[USER_CONFIG1_WIDTH–1:0]	user_config1	Sets the state of the **user_config1[USER_CONFIG1_WIDTH–1:0]** outputs

[a] If USER_CONFIG1_WIDTH is configured as 32 then none of the bits in this register are undefined.

3.3.31 Feature Control Register

The feature_ctrl Register characteristics are:

Purpose Controls the:

- early write response behavior
- write burst behavior.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-38 shows the feature_ctrl Register bit assignments.

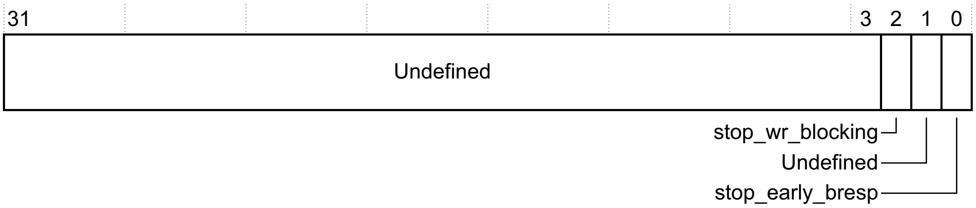


Figure 3-38 feature_ctrl Register bit assignments

Table 3-32 shows the feature_ctrl Register bit assignments.

Table 3-32 feature_ctrl Registers bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero
[2]	stop_wr_blocking	Controls the write merge up to memory burst: 0 = write transfers initiated when a full write memory burst is ready 1 = write transfers initiated when write data is available
[1]	-	Read undefined, write as zero
[0]	stop_early_bresp	Controls the early write response feature: 0 = early write response enabled 1 = early write response disabled

3.3.32 ECC Control Register

The ecc_control Register characteristics are:

Purpose Controls the:

- enabling of the *single-error correct, double-error detect* (SECDED) behavior
- enabling of interrupts.

Usage constraints No usage constraints.

Configurations Available when a DDR2 DMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-39 shows the ecc_control Register bit assignments.

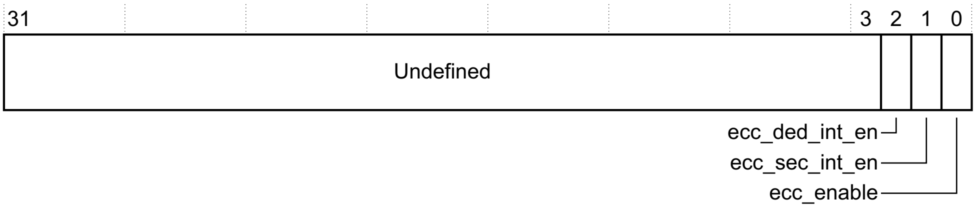


Figure 3-39 ecc_control Register bit assignments

Table 3-33 shows the ecc_control Register bit assignments.

Table 3-33 ecc_control Registers bit assignments

Bits	Name	Function
[31:3]	-	Undefined, write as zero.
[2]	ecc_ded_int_en	Controls the enabling of the double-error detect interrupt: 0 = disabled, the DDR2 DMC sets **ecc_ded_int** LOW. This is the default. 1 = enabled, the ECC block can modify the state of **ecc_ded_int**.
[1]	ecc_sec_int_en	Controls the clearing of the ECC single-error correct signal. Write: 0 = disabled, the DDR2 DMC sets **ecc_sec_int** LOW. This is the default. 1 = enabled, the ECC block can modify the state of **ecc_sec_int**.
[0]	ecc_enable	Controls the enabling of the SECDED ECC behavior: 0 = disabled 1 = enabled.

3.3.33 ECC Interrupt Clear Register

The ecc_int_clr Register characteristics are:

Purpose Controls the clearing of interrupts.

Usage constraints No usage constraints.

Configurations Available when a DDR2 DMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-40 on page 3-33 shows the ecc_int_clr Register bit assignments.

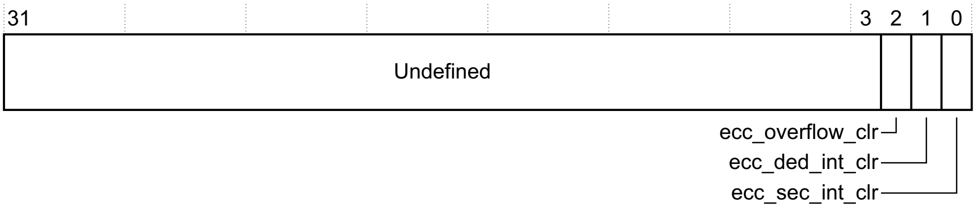


Figure 3-40 ecc_int_clr Register bit assignments

Table 3-34 shows the ecc_int_clr Register bit assignments.

Table 3-34 ecc_int_clr Registers bit assignments

Bits	Name	Function
[31:3]	-	Undefined, write as zero.
[2]	ecc_overflow_clr	Controls the clearing of the ECC overflow signal. Write: 0 = state of **ecc_overflow_int** is unchanged 1 = DDR2 DMC sets **ecc_overflow_int** LOW.
[1]	ecc_ded_int_clr	Controls the clearing of the ECC double-error detect signal. Write: 0 = state of **ecc_ded_int** is unchanged 1 = DDR2 DMC sets **ecc_ded_int** LOW.
[0]	ecc_sec_int_clr	Controls the clearing of the ECC single-error correct signal. Write: 0 = state of **ecc_sec_int** is unchanged 1 = DDR2 DMC sets **ecc_sec_int** LOW.

3.3.34 ECC Status Register

The ecc_status Register characteristics are:

Purpose Returns the status of the ECC interrupts.

Usage constraints No usage constraints.

Configurations Available when a DDR2 DMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-41 shows the ecc_status Register bit assignments.

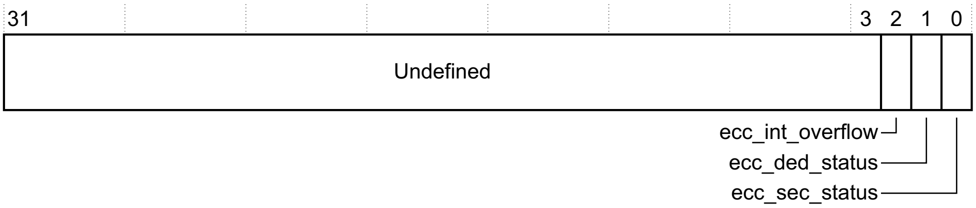


Figure 3-41 ecc_status Register bit assignments

Table 3-35 shows the ecc_status Register bit assignments.

Table 3-35 ecc_status Registers bit assignments

Bits	Name	Function
[31:3]	-	Undefined.
[2]	ecc_int_overflow	Returns the status of the ECC overflow signal: 0 = **ecc_overflow_int** is LOW 1 = **ecc_overflow_int** is HIGH.
[1]	ecc_ded_status	Returns the status of the double-error detect signal: 0 = **ecc_ded_int** is LOW 1 = **ecc_ded_int** is HIGH.
[0]	ecc_sec_status	Returns the status of the single-error correct signal: 0 = **ecc_sec_int** is LOW 1 = **ecc_sec_int** is HIGH.

3.3.35 ECC Information Register

The ecc_info0 Register characteristics are:

Purpose Returns the address location that contains either:

- data with a non-correctable double bit-error
- valid data, after the DDR2 DMC corrected a single bit-error.

Usage constraints No usage constraints.

Configurations Available when a DDR2 DMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-6.

Figure 3-42 shows the ecc_info0 Register bit assignments.

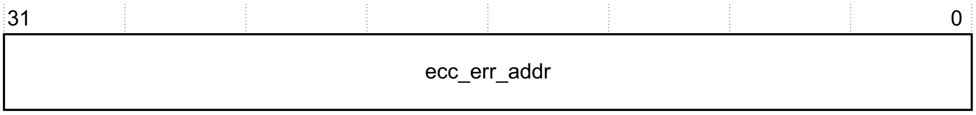


Figure 3-42 ecc_info0 Register bit assignments

Table 3-36 shows the ecc_info0 Register bit assignments.

Table 3-36 ecc_info0 Registers bit assignments

Bits	Name	Function
[31:0]	ecc_err_addr	Returns the address where the DDR2 DMC either: • detected a double bit-error • corrected a single bit-error.

3.3.36 Peripheral Identification Registers

The periph_id_[3:0] Register characteristics are:

Purpose Provide information about the configuration and version of the peripheral.

Usage constraints No usage constraints.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

These registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value. Figure 3-43 shows the correspondence between bits [7:0] of the periph_id Registers and the conceptual 32-bit Peripheral ID Register.

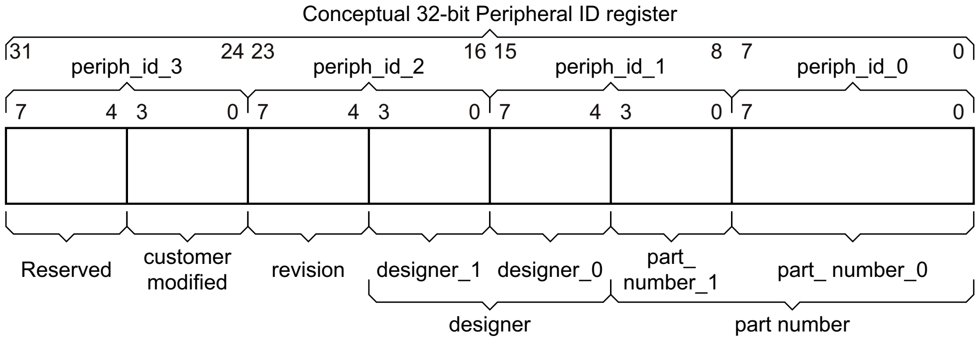


Figure 3-43 periph_id_[3:0] Register bit assignments

Table 3-37 shows the register bit assignments for the conceptual 32-bit peripheral ID register.

Table 3-37 Conceptual peripheral ID register bit assignments

Bits	Name	Function
[31:28]	-	Reserved.
[27:24]	customer modified	Identifies data that is relevant to an ARM partner.
[23:20]	revision	Identifies the RTL revision of the peripheral.
[19:12]	designer	Identifies the designer. This is `0x41` for ARM.
[11:0]	part_number	Identifies the peripheral. The part number for the DDR2 DMC is `0x341`.

The following sections describe the periph_id Registers:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1* on page 3-36
- *Peripheral Identification Register 2* on page 3-36
- *Peripheral Identification Register 3* on page 3-36.

Peripheral Identification Register 0

The periph_id_0 Register is hard-coded and the fields in the register set the reset value. Table 3-38 shows the register bit assignments.

Table 3-38 periph_id_0 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	part_number_0	Returns `0x41`

Peripheral Identification Register 1

The periph_id_1 Register is hard-coded and the fields in the register set the reset value. Table 3-39 shows the register bit assignments.

Table 3-39 periph_id_1 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined
[7:4]	designer_0	Returns `0x1`
[3:0]	part_number_1	Returns `0x3`

Peripheral Identification Register 2

The periph_id_2 Register is hard-coded and the fields in the register set the reset value. Table 3-40 shows the register bit assignments.

Table 3-40 periph_id_2 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined
[7:4]	revision	These bits return the revision number: `0x0` = r0p0 `0x1` = r0p1_00rel0 `0x2` = r0p1_01rel0 `0x3` = r1p0 `0x4` = r1p1
[3:0]	designer_1	Returns `0x4`

Peripheral Identification Register 3

The periph_id_3 Register is hard-coded and the fields in the register set the reset value. Table 3-41 shows the register bit assignments.

Table 3-41 periph_id_3 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined
[7:4]	-	Reserved for future use, read undefined
[3:0]	customer modified	Customer modified number, `0x0` from ARM

3.3.37 CoreLink Identification Registers

The pcell_id_[3:0] Register characteristics are:

Purpose When concatenated, these four registers return `0xB105F00D` to indicate that the DDR2 DMC is a CoreLink peripheral.

Usage constraints No usage constraints.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-6.

These registers can be treated conceptually as a single register that holds a 32-bit CoreLink identification value. You can use the register for automatic BIOS configuration.

Figure 3-44 shows the register bit assignments.

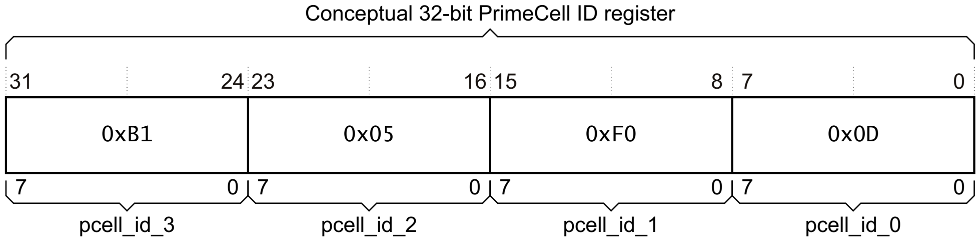


Figure 3-44 pcell_id Register bit assignments

Table 3-42 shows the register bit assignments.

Table 3-42 pcell_id Register bit assignments

pcell_id Register		pcell_id_[3:0] Registers		
Bits	**Reset value**	**Register**	**Bits**	**Description**
[31:24]	`0xB1`	pcell_id_3	[31:8]	Read undefined
			[7:0]	Returns `0xB1`
[23:16]	`0x05`	pcell_id_2	[31:8]	Read undefined
			[7:0]	Returns `0x05`
[15:8]	`0xF0`	pcell_id_1	[31:8]	Read undefined
			[7:0]	Returns `0xF0`
[7:0]	`0x0D`	pcell_id_0	[31:8]	Read undefined
			[7:0]	Returns `0x0D`

Chapter 4
Programmers Model for Test

This chapter describes the CoreLink DDR2 DMC test registers. It contains the following section:

4.1 Integration test registers

Test registers are provided for integration testing.

Figure 4-1 shows the integration test register map.

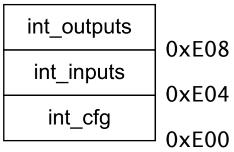


Figure 4-1 Integration test register map

Table 4-1 shows the integration test registers in base offset order.

Table 4-1 DDR2 DMC test register summary

Offset	Name	Type	Reset	Description
`0xE00`	int_cfg	RW	`0x0`	*Integration Configuration Register* on page 4-3
`0xE04`	int_inputs	RO	-[a]	*Integration Inputs Register* on page 4-4
`0xE08`	int_outputs	WO	-	*Integration Outputs Register* on page 4-5

[a] Dependent on the state of various input signals.

4.1.1 Integration Configuration Register

The int_cfg Register characteristics are:

Purpose Controls the enabling of the integration test logic.

Usage constraints Only accessible in Config state. ARM recommends that it is only accessed for integration testing or production testing.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 4-1 on page 4-2.

Figure 4-2 shows the int_cfg Register bit assignments.

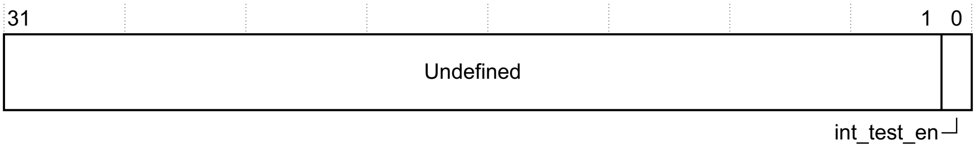


Figure 4-2 int_cfg Register bit assignments

Table 4-2 shows the int_cfg Register bit assignments.

Table 4-2 int_cfg Register bit assignments

Bits	Name	Function
[31:1]	-	Read undefined, write as zero.
[0]	int_test_en	Enables the integration test logic: 0 = disables the integration test logic 1 = enables the integration test logic. **Note** When the controller exits integration test mode, the **cactive** and **csysack** signals must be HIGH, even if the SoC does not use these signals. To satisfy this requirement you must program the int_outputs Register, see *Integration Outputs Register* on page 4-5.

4.1.2 Integration Inputs Register

The int_inputs Register characteristics are:

Purpose Provides the status of the following inputs:

- **csysreq**
- **qos_override[15:0]**.

Usage constraints

- Only accessible in Config state.
- Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. To enable the integration test logic see *Integration Configuration Register* on page 4-3.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 4-1 on page 4-2.

Figure 4-3 shows the int_inputs Register bit assignments.

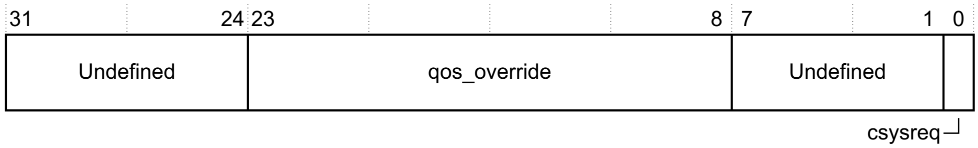


Figure 4-3 int_inputs Register bit assignments

Table 4-3 shows the int_inputs Register bit assignments.

Table 4-3 int_inputs Register bit assignments

Bits	Name	Function
[31:24]	-	Read undefined
[23:8]	qos_override	Returns the status of the **qos_override[15:0]** inputs
[7:1]	-	Read undefined
[0]	csysreq	Returns the status of the **csysreq** input

4.1.3 Integration Outputs Register

The int_outputs Register characteristics are:

Purpose Enables an external master to control the state of the following outputs:

- **cactive**
- **csysack**
- **ecc_sec_int**
- **ecc_ded_int**
- **ecc_overflow_int**.

Usage constraints

- Only accessible in Config state.
- Some bits are only present when a DDR2 DMC supports *Error Correction Code* (ECC).
- Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. To enable the integration test logic see *Integration Configuration Register* on page 4-3.

Configurations Available in all configurations of the DDR2 DMC.

Attributes See the register summary in Table 4-1 on page 4-2.

Figure 4-4 shows the int_outputs Register bit assignments.

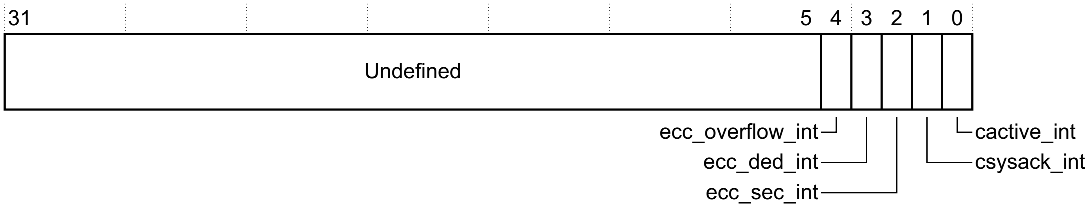


Figure 4-4 int_outputs Register bit assignments

Table 4-4 shows the int_outputs Register bit assignments.

Table 4-4 int_outputs Register bit assignments

Bits	Name	Function
[31:5]	-	Write as zero
[4]	ecc_overflow_int	Controls the state of the **ecc_overflow_int** output if the DDR2 DMC supports ECC
[3]	ecc_ded_int	Controls the state of the **ecc_ded_int** output if the DDR2 DMC supports ECC
[2]	ecc_sec_int	Controls the state of the **ecc_sec_int** output if the DDR2 DMC supports ECC
[1]	csysack_int	Controls the state of the **csysack** output
[0]	cactive_int	Controls the state of the **cactive** output

Appendix A
Signal Descriptions

This appendix describes the CoreLink DDR2 DMC signals. It contains the following sections:

A.1 Clock and reset signals

Table A-1 shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Type	Source	Description
aclk	Input	Clock source	Clock for the **aclk** domain.
aresetn	Input	Reset source	**aclk** domain reset signal. This signal is active LOW.
cclken	Input	Bus clock	Clock enable for the AXI low-power interface.
mclk	Input	Clock source	Clock for **mclk** domain.
mclkn[a]	Input	Clock source	Optional clock for the legacy pad interface.
mclkx2[a]	Input	Clock source	Optional clock for the legacy pad interface.
mclkx2n[a]	Input	Clock source	Optional clock for the legacy pad interface.
mresetn	Input	Reset source	Reset for **mclk** domain. This signal is active LOW.
pclken	Input	Bus clock	Clock enable for the APB interface.

[a] This signal is not available when the DDR2 DMC is configured to implement a *DDR PHY Interface* (DFI) pad interface.

A.2 Miscellaneous signals

This section describes the following miscellaneous signals:

- *QoS*
- *Tie-offs*
- *asetbok signal* on page A-4
- *User signals* on page A-4
- *Interrupt signals* on page A-4
- *Scan test* on page A-5.

A.2.1 QoS

Table A-2 shows the QoS signal.

Table A-2 QoS signal

Signal	Type	Source	Description
qos_override[15:0]	Input	External control logic	When one or more bits are HIGH, coincident with **arvalid** and **arready**, and when the **arid** match bits are equivalent to the **qos_override** bit(s), then the QoS for the read access is forced to minimum latency. See *Quality of Service* on page 2-16 for more information.

A.2.2 Tie-offs

Table A-3 shows the tie-off signals.

Table A-3 Tie-off signals

Signal	Type	Source	Description
bank_bits[1:0]	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the bank_bits field in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-22 for information about the values that a configured controller permits.
cke_init	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the cke_init bit in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-22.
dqm_init	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the dqm_init bit in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-22. This signal is only available when the DDR2 DMC is configured to provide a legacy pad interface.
memory_width[1:0]	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the memory_width2 field in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-22 for information about the values that a configured controller permits.
sync	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the clock_cfg [0] bit in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-22. Set this signal HIGH if **aclk** is synchronous to **mclk**. Set this signal LOW if **aclk** is asynchronous to **mclk**.

A.2.3 **asetbok** signal

Table A-4 shows the **asetbok** signal.

Table A-4 **asetbok** signal

Signal	Type	Destination	Description
asetbok	Output	DLL	Flags safe cycles to update the DLL control signals. This signal is only available when the DDR2 DMC is configured to provide a legacy pad interface.

A.2.4 User signals

These are general purpose I/O signals that you can use to control external devices, such as a *Delay-Locked Loop* (DLL). Table A-5 shows the user signals.

Table A-5 User signals

Signal	Type	Source or destination	Description
user_config0[USER_CONFIG0_WIDTH–1:0]	Output	External control logic	General purpose control signals that you program using the *User Config 0 Register* on page 3-30.
user_config1[USER_CONFIG1_WIDTH–1:0]	Output	External control logic	General purpose control signals that you program using the *User Config 1 Register* on page 3-30.
user_status[USER_STATUS_WIDTH–1:0]	Input	External control logic	General purpose input signals that are read using the *User Status Register* on page 3-29.

A.2.5 Interrupt signals

Table A-6 shows the ECC interrupt signals when a DDR2 DMC is configured to support ECC.

Table A-6 ECC interrupt signals

Signal	Type	Destination	Description
ecc_ded_int	Output	Interrupt controller	ECC double-error detect signal. If the DDR2 DMC detects a non-correctable double-error then is sets this signal HIGH.
ecc_overflow_int	Output	Interrupt controller	ECC overflow signal. If the DDR2 DMC attempts to set **ecc_ded_int** or **ecc_sec_int** HIGH but that signal is already HIGH then it sets this overflow signal HIGH.
ecc_sec_int	Output	Interrupt controller	ECC single-error correct signal. If the DDR2 DMC corrects a single bit-error then is sets this signal HIGH.

A.2.6 Scan test

Table A-7 shows the scan test signals.

Table A-7 Scan test signals

Signal	Type	Source	Description
dft_en_clk_out	Input	Tie-off	This signal is used for *Automatic Test Pattern Generator* (ATPG) testing only but is only available when the DDR2 DMC is configured to provide a legacy pad interface
rst_bypass	Input	Tie-off	This signal is used for ATPG testing only
se	Input	Tie-off	This signal enables scan mode

A.3 AXI signals

The following sections list the AXI signals:

- *Write address channel signals*
- *Write data channel signals*
- *Write response channel signals* on page A-7
- *Read address channel signals* on page A-7
- *Read data channel signals* on page A-8
- *AXI low-power interface signals* on page A-8.

A.3.1 Write address channel signals

Table A-8 shows the AXI write address channel signals.

Table A-8 Write address channel signals

Signal	AMBA equivalent[a]
awaddr[31:0]	**AWADDR**
awburst[1:0]	**AWBURST[1:0]**
awcache[3:0][b]	**AWCACHE[3:0]**
awid[AID_WIDTH–1:0][c]	**AWID**
awlen[3:0]	**AWLEN[3:0]**
awlock[1:0]	**AWLOCK[1:0]**
awprot[2:0][b]	**AWPROT[2:0]**
awready	**AWREADY**
awsize[2:0]	**AWSIZE[2:0]**
awvalid	**AWVALID**

[a] See the *AMBA AXI Protocol Specification* for a description of these signals.

[b] The DDR2 DMC ignores any information that it receives on these signals.

[c] The value of **AID_WIDTH** is set during configuration of the DDR2 DMC.

A.3.2 Write data channel signals

Table A-9 shows the AXI write data channel signals.

Table A-9 Write data channel signals

Signal	AMBA equivalent[a]
wdata[AXI_DATA_MSB:0][b]	**WDATA**
wid[AID_WIDTH–1:0][b]	**WID**
wlast	**WLAST**
wready	**WREADY**
wstrb[AXI_STRB_MSB:0][b]	**WSTRB**
wvalid	**WVALID**

a. See the *AMBA AXI Protocol Specification* for a description of these signals.
b. The value of **AXI_DATA_MSB** and **AID_WIDTH** are set during configuration of the DDR2 DMC. **AXI_STRB_MSB**=**AXI_DATA_MSB**÷8.

A.3.3 Write response channel signals

Table A-10 shows the AXI write response channel signals.

Table A-10 Write response channel signals

Signal	AMBA equivalent[a]
bid[AID_WIDTH–1:0][b]	**BID**
bready	**BREADY**
bresp[1:0][c]	**BRESP[1:0]**
bvalid	**BVALID**

a. See the *AMBA AXI Protocol Specification* for a description of these signals.
b. The value of **AID_WIDTH** is set during configuration of the DDR2 DMC.
c. The DDR2 DMC ties **bresp[1]** LOW and therefore it only provides OKAY or EXOKAY responses.

A.3.4 Read address channel signals

Table A-11 shows the AXI read address channel signals.

Table A-11 Read address channel signals

Signal	AMBA equivalent[a]
araddr[31:0]	**ARADDR**
arburst[1:0]	**ARBURST[1:0]**
arcache[3:0][b]	**ARCACHE[3:0]**
arid[AID_WIDTH–1:0][c]	**ARID**
arlen[3:0]	**ARLEN[3:0]**
arlock[1:0]	**ARLOCK[1:0]**
arprot[2:0][b]	**ARPROT[2:0]**
arready	**ARREADY**
arsize[2:0]	**ARSIZE[2:0]**
arvalid	**ARVALID**

a. See the *AMBA AXI Protocol Specification* for a description of these signals.
b. The DDR2 DMC ignores any information that it receives on these signals.
c. The value of **AID_WIDTH** is set during configuration of the DDR2 DMC.

A.3.5 Read data channel signals

Table A-12 shows the AXI read data channel signals.

Table A-12 Read data channel signals

Signal	AMBA equivalent[a]
rdata[AXI_DATA_MSB:0][b]	**RDATA**
rid[AID_WIDTH–1:0][b]	**RID**
rlast	**RLAST**
rready[c]	**RREADY**
rresp[1:0][d]	**RRESP[1:0]**
rvalid	**RVALID**

a. See the *AMBA AXI Protocol Specification* for a description of these signals.

b. The value of **AXI_DATA_MSB** and **AID_WIDTH** are set during configuration of the DDR2 DMC.

c. It is possible for refreshes to be missed if **rready** is held LOW for longer than one refresh period, and the read data FIFO, command FIFO, and arbiter queue become full. An OVL error is triggered if this occurs in simulation. Ensure that the device has a sufficiently high system priority to prevent this.

d. The DDR2 DMC ties **rresp[1]** LOW and therefore it only provides OKAY or EXOKAY responses.

A.3.6 AXI low-power interface signals

Table A-13 shows the AXI low-power interface signals.

Table A-13 AXI low-power interface signals

Signal	AMBA equivalent[a]
cactive	**CACTIVE**
csysack	**CSYSACK**
csysreq	**CSYSREQ**

a. See the *AMBA AXI Protocol Specification* for a description of these signals.

A.4 APB signals

Table A-14 shows the APB signals.

Table A-14 APB interface signals

Signal	AMBA equivalent[a]
paddr[31:0][b]	**PADDR**
penable	**PENABLE**
prdata[31:0]	**PRDATA**
pready	**PREADY**
psel	**PSELx**
pslverr[c]	**PSLVERR**
pwdata[31:0]	**PWDATA**
pwrite	**PWRITE**

[a] See the *AMBA 3 APB Protocol Specification* for a description of these signals.

[b] The DDR2 DMC uses bits [11:2]. It ignores bits [31:12] and bits [1:0].

[c] The DDR2 DMC ties **pslverr** LOW.

A.5 Pad interface signals

The DDR2 DMC can implement either a legacy pad interface or a DFI pad interface. The choice of pad interface is set during configuration of the DDR2 DMC.

The following sections describe:

- *Legacy pad interface*
- *DFI pad interface* on page A-11.

A.5.1 Legacy pad interface

Table A-15 shows the legacy pad interface signals.

Table A-15 Legacy pad interface signals

Signal	Type	Source or destination	Description
Signals for a DDR2 DMC, with or without ECC support:			
add[15:0]	Output	External memory	Address
ap	Output	External memory	Auto precharge
ba[MEMORY_BANK_WIDTH–1:0][a]	Output	External memory	Bank select
cas_n	Output	External memory	Column address strobe
cke	Output	External memory	Clock enable
clk_out[MEMORY_CHIPS–1:0][b]	Output	External memory	Memory clock
cs_n[MEMORY_CHIPS–1:0]	Output	External memory	Chip select
data_en	Output	External memory	Data direction enable
dqm[MEMORY_BYTES–1:0][c]	Output	External memory	Data bus mask
dqs_in_<n>	Input	External memory	Data strobe in
dqs_in_n_<n>	Input	External memory	Data strobe in bar
dqs_out_<n>	Output	External memory	Data strobe out
odt[MEMORY_CHIPS–1:0]	Output	External memory	Memory IO termination control
odt_read	Output	Output pad IO	Device IO termination control
ras_n	Output	External memory	Row address strobe
we_n	Output	External memory	Write enable
Signals for a DDR2 DMC without ECC support:			
dq_in[MEMWIDTH–1:0][d]	Input	External memory	Data bus in
dq_out[MEMWIDTH–1:0][d]	Output	External memory	Data bus out

Table A-15 Legacy pad interface signals (continued)

Signal	Type	Source or destination	Description
Signals for a DDR2 DMC with ECC support:			
dq_in[MEMWIDTH+ECCWIDTH–1:0][e]	Input	External memory	Data bus in
dq_out[MEMWIDTH+ECCWIDTH–1:0][e]	Output	External memory	Data bus out
dqs_in_ecc	Input	External memory	ECC strobe in
dqs_in_n_ecc	Input	External memory	ECC strobe in bar
dqs_out_ecc	Output	External memory	ECC strobe out

[a] The MEMORY_BANK_WIDTH value depends on the number of banks in the configuration.

[b] MEMORY_CHIPS is the number of chip selects and is set during configuration of the DDR2 DMC.

[c] MEMORY_BYTES is the width of the external memory data bus in bytes and is set during configuration of the DDR2 DMC.

[d] MEMWIDTH is the width of the external memory data bus in bits and is set during configuration of the DDR2 DMC.

[e] The value of ECCWIDTH depends on MEMWIDTH as follows:
- if MEMWIDTH = 16 then ECCWIDTH = 6
- if MEMWIDTH = 32 then ECCWIDTH = 7
- if MEMWIDTH = 64 then ECCWIDTH = 8.

A.5.2 DFI pad interface

Table A-16 shows the DFI pad interface signals. See the *DDR PHY Interface (DFI) Specification* for a description of these signals.

Table A-16 DFI pad interface signals

Signal	Type	Source or destination
Signals for a DDR2 DMC, with or without ECC support:		
dfi_address[15:0]	Output	PHY device
dfi_bank[MEMORY_BANK_WIDTH–1:0][a]	Output	
dfi_cas_n	Output	
dfi_cke[MEMORY_CHIPS–1:0][b]	Output	
dfi_cs_n[MEMORY_CHIPS–1:0][b]	Output	
dfi_dram_clk_disable[MEMORY_CHIPS–1:0][b]	Output	
dfi_init_complete	Input	
dfi_odt[MEMORY_CHIPS–1:0][b]	Output	
dfi_phyupd_ack	Output	
dfi_phyupd_req	Input	
dfi_phyupd_type[1:0]	Input	
dfi_ras_n	Output	
dfi_we_n	Output	

Table A-16 DFI pad interface signals (continued)

Signal	Type	Source or destination
Signals for a DDR2 DMC without ECC support:		
dfi_rddata[2×MEMWIDTH–1:0][c]	Input	PHY device
dfi_rddata_en[2×MEMORY_BYTES–1:0][d]	Output	
dfi_wrdata_en[2×MEMORY_BYTES–1:0][d]	Output	
dfi_wrdata[2×MEMWIDTH–1:0][c]	Output	
dfi_wrdata_mask[2×MEMORY_BYTES–1:0][d]	Output	
Signals for a DDR2 DMC with ECC support:		
dfi_rddata[2×(MEMWIDTH+ECCWIDTH)–1:0][c e]	Input	PHY device
dfi_rddata_en[2×(MEMORY_BYTES+1)–1:0][d]	Output	
dfi_wrdata_en[2×(MEMORY_BYTES+1)–1:0][d]	Output	
dfi_wrdata[2×(MEMWIDTH+ECCWIDTH)–1:0][c e]	Output	
dfi_wrdata_mask[2×(MEMORY_BYTES+1)–1:0][d]	Output	

[a] The MEMORY_BANK_WIDTH value depends on the number of banks in the configuration.

[b] MEMORY_CHIPS is the number of chip selects and is set during configuration of the DDR2 DMC.

[c] MEMWIDTH is the width of the external memory data bus in bits and is set during configuration of the DDR2 DMC.

[d] MEMORY_BYTES is the width of the external memory data bus in bytes and is set during configuration of the DDR2 DMC.

[e] The value of ECCWIDTH depends on MEMWIDTH as follows:
- if MEMWIDTH = 16 then ECCWIDTH = 6
- if MEMWIDTH = 32 then ECCWIDTH = 7
- if MEMWIDTH = 64 then ECCWIDTH = 8.

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Differences between issue C and issue D

Change	Location	Affects
Added requirement to issue an `AUTO REFRESH` command before the DDR2 DMC moves from the Config state to the Paused state	*Memory manager* on page 2-17	All revisions
Updated the clock_cfg field description	Table 3-21 on page 3-23	All revisions
Updated the write data value for the following registers: • t_xsr • t_esr • memory_cfg2	Table 2-3 on page 2-26	All revisions
Updated the memory device initialization example	*DDR2 device initialization* on page 2-27	All revisions
Updated the reset value of the memory_cfg Register from `0x00010020` to `0x00010021`	Table 3-1 on page 3-6	r1p0
Updated the reset value of the following registers: • t_rfc Register from `0x00002123` to `0x00002023` • t_rcd and t_rp Registers from `0x00000305` to `0x00000205` • t_faw Register from `0x00000011` to `0x00001114`		All revisions
Updated the reset value of the refresh_prd Register from `0x00000A2C` to `0x000001E7`		r0p1

Table B-1 Differences between issue C and issue D (continued)

Change	Location	Affects
Added conditions about changing the power_dwn_prd bit	*Memory Configuration Register* on page 3-12	All revisions
Updated the function of the row_bits field when set to `0b000` or `0b001`		
Updated the supported values for the cas_latency field from 2-6 to 3-6	*CAS Latency Register* on page 3-14	All revisions
Increased the bit widths for the schedule_rfc and rfc fields	*AUTO REFRESH to Command Timing Register* on page 3-18	r1p0
Added usage constraints for writes	*Exit Power-down Timing Register* on page 3-20	All revisions
Changed memory_width field to memory_width2	*Memory Configuration 2 Register* on page 3-22	All revisions
Updated the most significant byte of the conceptual peripheral ID register	• Figure 3-43 on page 3-35 • Table 3-37 on page 3-35	All revisions
Added requirement to set **cactive** and **csysack** HIGH when the controller exits integration test mode	*Integration Configuration Register* on page 4-3	All revisions
Added an _int suffix to the cactive and csysack bits	*Integration Outputs Register* on page 4-5	All revisions
Added configurable bus width for the **user_status**, **user_config0**, and **user_config1** signals	Throughout book	r1p0

Table B-2 Differences between issue D and issue E

Change	Location	Affects
Updated information on exclusive access monitor	*Exclusive access* on page 2-11	All revisions
Added `0x4` to the revision field	*Peripheral Identification Register 2* on page 3-36	r1p1
Updated description of the memc_status field	*Memory Controller Status Register* on page 3-8	r1px

Glossary

This glossary describes some of the terms used in technical documents from ARM.

Advanced eXtensible Interface (AXI)

A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced Microcontroller Bus Architecture (AMBA)

A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)

A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

Architecture The organization of hardware and/or software that characterizes a processor and its attached components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture, ARMv6 architecture.

ATPG *See* Automatic Test Pattern Generation.

Automatic Test Pattern Generation (ATPG)

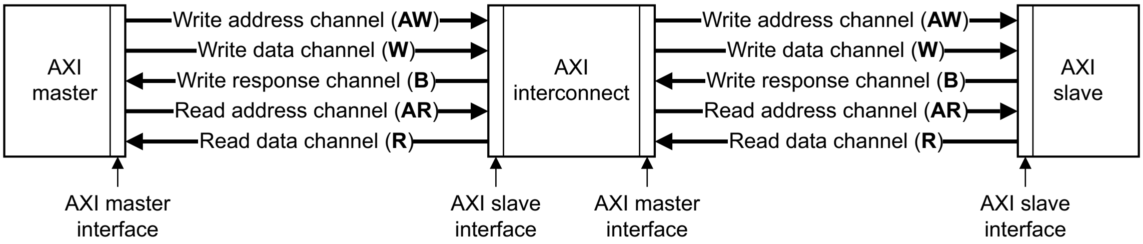
The process of automatically generating manufacturing test vectors for an ASIC design, using a specialized software tool.

AXI *See* Advanced eXtensible Interface.

AXI channel order and interfaces

The block diagram shows:

- the order in which AXI channel signals are described
- the master and slave interface conventions for AXI components.



AXI terminology The following AXI terms are general. They apply to both masters and slaves:

Active read transaction

A transaction for which the read address has transferred, but the last read data has not yet transferred.

Active transfer

A transfer for which the **xVALID**[1] handshake has asserted, but for which **xREADY** has not yet asserted.

Active write transaction

A transaction for which the write address or leading write data has transferred, but the write response has not yet transferred.

Completed transfer

A transfer for which the **xVALID**/**xREADY** handshake is complete.

Payload The non-handshake signals in a transfer.

Transaction An entire burst of transfers, comprising an address, one or more data transfers and a response transfer (writes only).

Transmit An initiator driving the payload and asserting the relevant **xVALID** signal.

[1] The letter **x** in the signal name denotes an AXI channel as follows:

AW	Write address channel.
W	Write data channel.
B	Write response channel.
AR	Read address channel.
R	Read data channel.

Transfer A single exchange of information. That is, with one **xVALID**/**xREADY** handshake.

The following AXI terms are master interface attributes. To obtain optimum performance, they must be specified for all components with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that a master interface can generate. It is specified for master interfaces that use combined storage for active write and read transactions. If not specified then it is assumed to be equal to the sum of the write and read issuing capabilities.

Read ID capability

The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that a master interface can generate.

Write ID capability

The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the master interface is capable of transmitting data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate.

The following AXI terms are slave interface attributes. To obtain optimum performance, they must be specified for all components with an AXI slave interface:

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. It is specified for slave interfaces that use combined storage for active write and read transactions. If not specified then it is assumed to be equal to the sum of the write and read acceptance capabilities.

Read acceptance capability

The maximum number of active read transactions that a slave interface can accept.

Read data reordering depth

The number of active read transactions for which a slave interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that a slave interface can accept.

Write interleave depth

The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.

Beat

Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

See also Burst.

Burst

A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

See also Beat.

Clock gating

Gating a clock signal for a macrocell with a control signal, and using the modified clock that results to control the operating state of the macrocell.

Halfword

A 16-bit data item.

Macrocell

A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as a processor, an ETM, and a memory block) plus application-specific logic.

Memory bank

One of two or more parallel divisions of interleaved memory, usually one word wide, that enable reads and writes of multiple words at a time, rather than single words. All memory banks are addressed simultaneously and a bank enable or chip select signal determines which of the banks is accessed for each transfer. Accesses to sequential word addresses cause accesses to sequential banks. This enables the delays associated with accessing a bank to occur during the access to its adjacent bank, speeding up memory transfers.

Processor

A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Region

A partition of instruction or data memory space.

Reserved

A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

Scan chain

A scan chain is made up of serially-connected devices that implement boundary scan technology using a standard JTAG TAP interface. Each device contains at least one TAP controller containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected parts of the device.

Unpredictable

For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.

Word

A 32-bit data item.