ARM® CoreLink™ DMC-500 Dynamic Memory Controller

Revision: r0p0

Technical Overview

ARM

ARM® CoreLink™ DMC-500 Dynamic Memory Controller

Technical Overview

Copyright © 2015, 2016 ARM. All rights reserved.

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	21 September 2015	Non-Confidential	First release of r0p0
0000-01	07 December 2015	Non-Confidential	Second release of r0p0
0000-02	13 January 2016	Non-Confidential	Third release of r0p0

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.

Unrestricted Access is an ARM internal classification.

Product Status

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

Web Address

http://www.arm.com

Contents
ARM® CoreLink™ DMC-500 Dynamic Memory Controller Technical Overview

Preface

This preface introduces the *ARM® CoreLink™ DMC-500 Dynamic Memory Controller Technical Overview*.

It contains the following:
-
-

About this book

This book is for the ARM® CoreLink™ DMC-500 Dynamic Memory Controller.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is intended for integrated circuit designers.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter describes the DMC-500.

Chapter 2 Functional Description
This chapter describes how the DMC-500 operates.

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

Glossary

The ARM Glossary is a list of terms used in ARM documentation, together with definitions for those terms. The ARM Glossary does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See the *ARM Glossary* for more information.

Typographic conventions

italic
Introduces special terminology, denotes 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.

<u>`mono`</u>`space`
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>`
Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

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

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure 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.
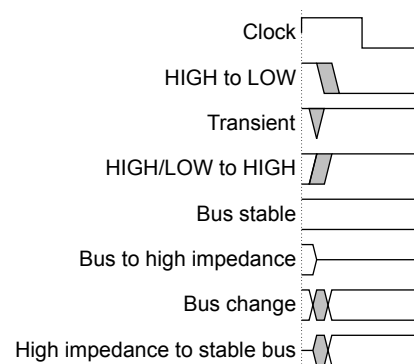
Figure 1 Key to timing diagram conventions

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

See *Infocenter*, *http://infocenter.arm.com*, for access to ARM documentation.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:
- *ARM® CoreLink™ DMC-500 Dynamic Memory Controller r0p0 Technical Reference Manual* (ARM 100132).
- *ARM® CoreLink™ DMC-500 Dynamic Memory Controller r0p0 Integration Manual* (ARM 100134).
- *ARM® CoreLink™ DMC-500 Dynamic Memory Controller r0p0 Implementation Guide* (ARM 100133).
- *ARM® Low Power Interface Specification, Q-Channel and P-Channel Interfaces* (ARM IHI 0068).
- *Principles of ARM® Memory Maps White Paper* (ARM DEN 0001).

Other publications

Feedback

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 and diagnostic procedures if appropriate.

Feedback on content

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

- The title *ARM® CoreLink™ DMC-500 Dynamic Memory Controller Technical Overview*.
- The number ARM 100131_0000_02_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

——————— **Note** ———————

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

———————————————

Chapter 1
Introduction

This chapter describes the DMC-500.

It contains the following sections:

1.1   About the product

This is a high-level overview of the DMC-500.

The DMC-500 is an ARM AMBA® SoC peripheral developed, tested, and licensed by ARM. It is a high-performance, area-optimized memory controller that is compatible with the AMBA 4 AXI protocol. It supports the following memory devices:

- *Low-Power Double Data Rate 3* (LPDDR3) SDRAM.
- LPDDR4 SDRAM.

The following figure shows an example system.
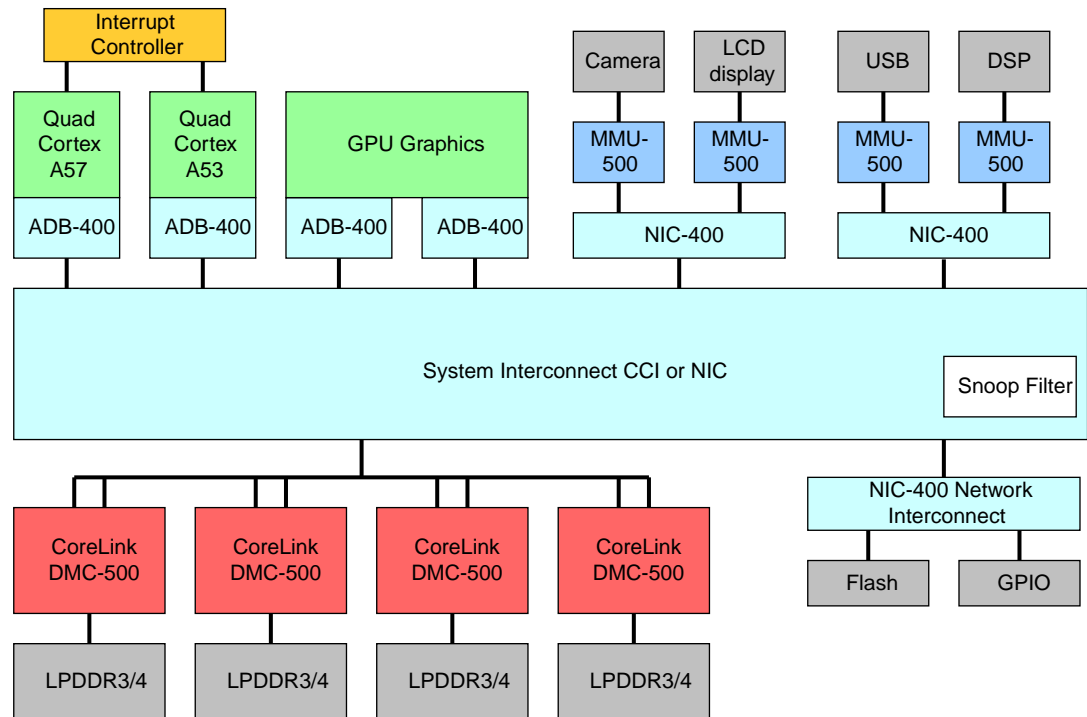
Figure 1-1 Example system

The DMC-500 enables data transfer between the SoC and the SDRAM devices external to the chip. It connects to the on-chip system through two System interfaces and to a processor through the Programming interface to program the DMC. It connects to the SDRAM devices through its memory interface block and the *DDR PHY Interface* (DFI).

1.2 Compliance

The DMC-500 is compatible with the following protocol specifications and standards:

- AMBA 4 AXI protocol.
- ARM Q-Channel and P-Channel Low-Power Interfaces.
- JEDEC LPDDR3 JESD209-3 standard.
- JEDEC LPDDR4 JESD209-4 standard.
- DFI 4.0.

1.3   Features

The DMC-500 supports the connection from AMBA system interfaces to off-chip SDRAMs. In addition, *Quality of Service* (QoS) features and ARM TrustZone® architecture security extensions are built in throughout the controller.

The DMC provides a System interface for a connection from the DMC to the CoreLink *Cache Coherent Interconnect* (CCI) or *Network Interconnect* (NIC). The programming interface enables configuration and initialization of the DMC-500.

The DMC-500 has the following features:

*   TrustZone architecture security extensions.
*   Buffering to optimize read and write turnaround and to maximize bandwidth.
*   System interfaces that provide:
    — An interface to connect to a network.
    — An interface for configuration and initialization purposes.
*   A *Memory Interface* (MI) that provides:
    — A DFI interface to a PHY.
*   Low-power operation through programmable SDRAM power modes.
*   Refresh Control Logic for memory banks.
*   Power Control Logic to generate power down requests to the SDRAM, and manage power enables for the PHY logic.

1.4   Interfaces

This section lists the interfaces in the DMC-500.

The DMC-500 has the following external interfaces:
- Two system interfaces to provide read and write access to or from a master.
- An AXI4 programming interface to program and control the DMC-500.
- A DFI4 PHY interface to transfer data to and from the external memory.
- A profile and debug interface.
- User I/O ports.
- A set of interrupts used to detect some operational events or handle errors for example.
- A low-power control interface that uses the P-channel protocol.
- A Q-channel interface for each AXI interface.

1.5 Configurable options

The `SYSTEM_SHUTTER_BOUNDARY` parameter is a parameter that you can override at instantiation time.

The `SYSTEM_SHUTTER_BOUNDARY` parameter controls the removed bits from the address when the system contains multiple memory channels or multiple DMCs. The range of this parameter is from 256-4096.

The `SYSTEM_SHUTTER_BOUNDARY` parameter is configured in number of bytes. For example, the default value of 4096 results in the addr_shutter bitfield effecting address bits 12, 13, and 14.

─────── **Note** ───────

The system must ensure that the DMC receives no access that crosses the configured boundary when this function is used.

─────────────────────

The `SYSTEM_ID_WIDTH` parameter has a value of 8-24 with a default value of 16. It configures the widths of the System interface ID signals.

The `SYNC` parameter specifies the relation between **s0clk**,**s1clk**, and **mclk**. It can have the following values:

0

Signifies that **s0clk**, **s1clk**, and **mclk** do not come from the same source.

1

Signifies the removal of the async crossings from **s0clk** and **s1clk** to **mclk**, because they come from the same source.

1.6 Test features

The DMC-500 provides the following test features:

- Integration test logic for integration testing.
- Debug and performance counters to monitor transaction events.

1.7   Product documentation and design flow

This section describes the DMC books and how they relate to the design flow.

Documentation

The DMC documentation is as follows:

Technical Overview
The *Technical Overview* (TO) summarizes the functionality of the DMC.

Technical Reference Manual
The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the DMC. It is required at all stages of the design flow. The choices made in the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the DMC then contact:

- The implementer to determine what integration, if any, was performed before implementing the DMC.
- The integrator to determine the pin configuration of the device that you are using.

The TRM is a confidential book that is only available to licensees.

Implementation Guide
The *Implementation Guide* (IG) describes:

- How to synthesize the *Register Transfer Level* (RTL).
- How to integrate RAM arrays.
- How to run test patterns.
- The processes to sign off the configured design.

The ARM product deliverables include reference scripts and information about using them to implement your design. Reference methodology flows supplied by ARM are example reference implementations. Contact your EDA vendor for EDA tool support.

The IG is a confidential book that is only available to licensees.

Integration Manual
The *Integration Manual* (IM) describes how to integrate the DMC into a SoC. It includes a description of the pins that the integrator must tie off to connect the DMC into an SoC design or to other IP.

The IM is a confidential book that is only available to licensees.

Design flow

The DMC is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following processes:

Implementation
The implementer synthesizes the RTL to produce a hard macrocell. This includes integrating RAMs into the design.

Integration
The integrator connects the implemented design into a SoC. This includes connecting it to a memory system.

Programming
This is the last process. The system programmer develops the software required to initialize the DMC, and tests the required application software.

Each process:

- Can be performed by a different party.
- Can include implementation and integration choices that affect the behavior and features of the DMC.

The operation of the final device depends on:

Configuration inputs

The integrator configures some features of the DMC by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software programming

The programmer configures the DMC by programming particular values into registers. This affects the behavior of the DMC.

————— **Note** —————

This manual refers to implementation-defined features. Reference to a feature that is included means that the appropriate pin configuration options are selected. Reference to an enabled feature means one that has also been configured by software.

—————————————

1.8 Product revisions

This section describes the differences in functionality between product revisions of the DMC-500.

r0p0

 First release.

Chapter 2
Functional Description

This chapter describes how the DMC-500 operates.

It contains the following sections:

2.1 About the functions

This section gives a brief description of all of the functions of the device.

The following figure shows a block diagram of the functions of the DMC-500. The colors show the different categories of functions:

- Blue indicates the blocks that are associated with data flow. The System interface is an example.
- Green indicates the blocks that are associated with programming. The Programming interface is an example.
- Orange indicates the blocks that are associated with the quality and efficiency of the communication to its external memory. The QoS engine is an example.
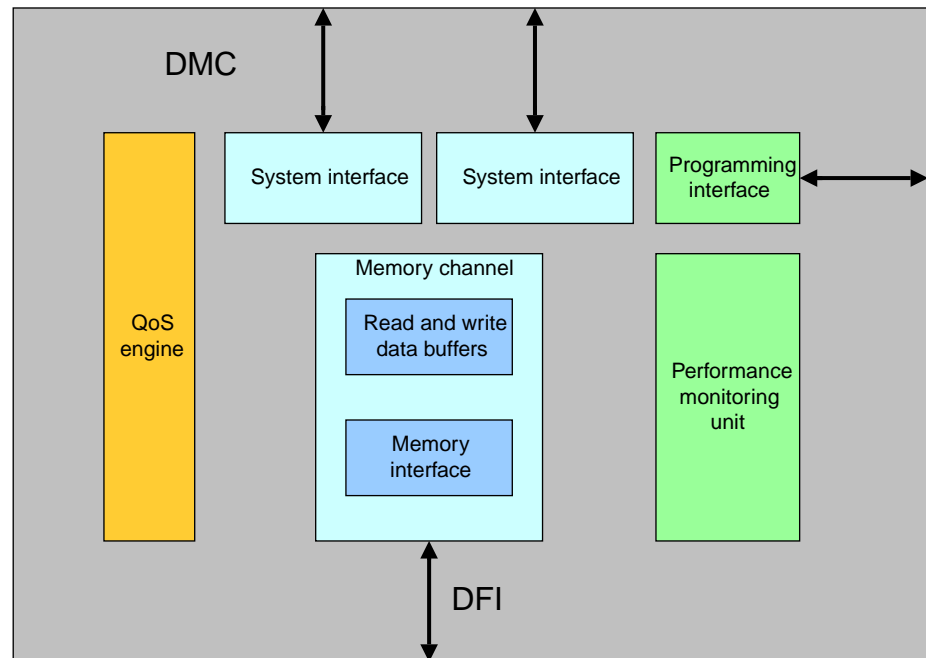
Figure 2-1 DMC functional block diagram

This section contains the following subsections:

2.1.1 System interface

The DMC-500 interfaces to the rest of the SoC through this interface.

For any attempted accesses that the system makes outside of the programmed address range of the DMC-500, the system interface responds with a non-data error response. According to how you program the DMC-500, it converts the system access information to the correct rank, bank, column, and row access of the external SDRAM that connects to it. The system interface supports TrustZone features to regulate Secure and Non-secure accesses to both Secure and Non-secure regions of memory

The DMC monitors queue occupancies and dictates whether system requests of any given QoS are to be accepted. Requests are allocated based on a threshold setting, derived from register settings.

2.1.2 Memory channel

Through this interface the DMC conducts its data transactions with the SDRAM and regulates the power consumption of the SDRAM.

2.1.3 Programming interface

Through this interface a master in the system programs the DMC.

You can define the Secure and Non-secure regions of external memory and also define how the DMC addresses the external memory from the address that the system provides on its system interface. You can also make direct accesses to the SDRAM, for example to initialize it.

2.1.4 Performance monitoring unit

You can use the *Performance Monitoring Unit* (PMU) counters to monitor the performance and power settings for your specific application.

These counters allow you to monitor the inner workings of the device and so enables additional information to be viewed.

2.1.5 QoS engine

The DMC provides controls to enable you to adjust its arbitration scheme for your system to maximize the availability of your external memory devices.

It provides buffers to re-order system transaction requests. It uses an advanced scheduling algorithm to ensure that traffic going to one memory bank causes minimal disruption to traffic going to a different memory bank. It also schedules transaction requests according to the availability of the destination memory bank. For system access requests to different available memory banks the DMC arbitrates these requests based on the QoS priority initially then on the temporal priority. These memory access requests all compete for control of the external SDRAM bus and SDRAM bank availability.

2.2 Clocking and resets

The DMC contains four clock domains. One for each system interface, one for the memory interface, and one for the programming interface. Each domain can be operated asynchronously from one another.

This section shows the clock and reset signals that the DMC requires.

This section contains the following subsections:

2.2.1 Clocks

The following requirements apply:

- **mclk** must run synchronously with, and at the same speed as, the PHY and SDRAM. **sclk** must run synchronously with, and at the same speed as, the interfacing system interconnect.
- **cfgclk**, **sclk0**, **sclk1**, and **mclk** can run asynchronously to each other, except when the SYNC parameter is asserted HIGH.

2.2.2 Resets

The DMC-500 has one reset input signal for each clock domain. They are **s0resetn**, **s1resetn**, **mresetn**, and **cfgresetn**.

The reset signals can be asserted asynchronously to their respective clock but must be deasserted synchronously to the positive edge of their respective clock.

——————— **Note** ———————

The DMC-500 contains reset repeaters so must be clocked for two cycles in reset to fully reset the device.

————————————————

The DMC-500 has the following usage restrictions:
1. You must always reset all domains. You must not assert the reset for one domain and then deassert that reset without asserting the reset for the other domain first.
2. You must deassert all resets before any AXI transactions arrive at the DMC.
3. When DMC is in the OFF state through the P-channel, there is no restriction on ordering of assertion and deassertion of resets.
4. When DMC is in the SLEEP state through software, the reset signals must be asserted in the following order: **mresetn**, **s0resetn** and **s1resetn**, then **cfgresetn**. There is no restriction on the order of the deassertion of resets.
5. ARM does not guarantee the correct function of the DMC if you apply resets when the DMC is not in either the OFF state or the SLEEP state.

——————— **Note** ———————

- To assert any DMC-500 reset signal, you must set it LOW.

————————————————

2.3 Interfaces

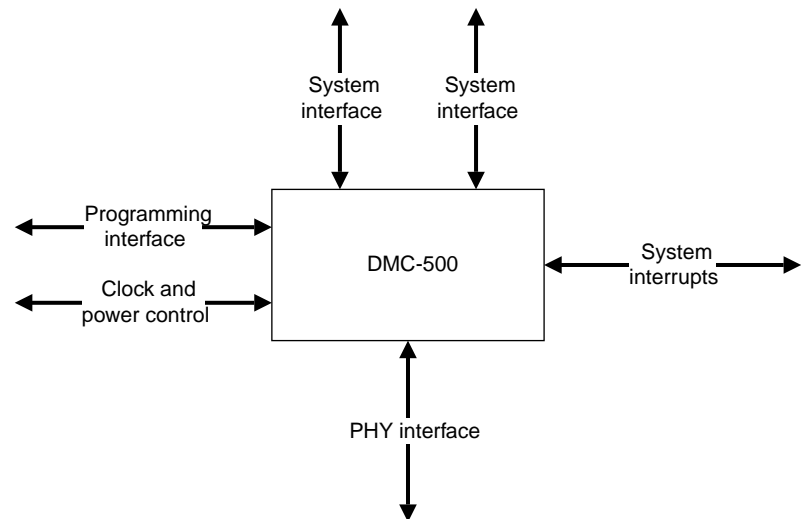This section describes the interfaces of the DMC-500, as the following figure shows.

Figure 2-2 Interfaces of the DMC

This section contains the following subsections:

2.3.1 System interface

The System Interface provides protocol conversion between the system and internal read/write requests.

2.3.2 Programming interface

Use this interface to program the DMC.

This AMBA slave interface allows software to configure the controller and to initialize the memory devices. It also provides a means of performing architectural state transitions and also querying certain debug and profile information. The interface is a memory-mapped register interface.

2.3.3 PHY interface

The PHY interface provides command scheduling and arbitration, including the generation of any required SDRAM prepare commands, for example, ACTIVATE and PRECHARGE. This section describes the PHY interface in the DMC-500.

The PHY interface is a DFI interface. It provides:
- Command scheduling and arbitration, including generation of any required SDRAM prepare commands, for example, ACTIVATE, or PRECHARGE.
- Automated AUTOREFRESH command generation.
- SDRAM interface link protection including automated retries for failed commands to ensure the correct ordering of those retried commands to SDRAM.
- Automated SDRAM and PHY logic power control.
- Profile and debug information.

2.3.4 Low-power control interface

This section describes the clock requirements for the DMC-500.

The DMC-500 provides a low-power control interface using the P-channel protocol. This is used to place the DMC into its low-power state, in this state the clock can be removed. The system can use the programming interface to do the following:

- Put the DMC into its low-power state.
- Take the DMC out of its low-power state.

P-Channel

The P-Channel is a simple power-controller-to-device interface that manages device power states.

The P-Channel interface has the following features:

- The power controller manages the power state transitions of the DMC.
- The DMC can optionally indicate a hint to the power controller for an opportunistic state transition.
- The DMC can deny a power state transition request.
- Robust clock domain crossing semantics enable safe asynchronous interfacing.

This protocol is a generic way to request a transition to a particular state using a request-acknowledge 4-phase handshake. The specific state transitions of the device under management are not restricted by the P-Channel protocol, but might be restricted by the capabilities of the device, as they are in the DMC.

The P-Channel contains the following signals, where * is an identifier for a power domain:

PREQ_* Indicates a request for a power state transition.

PSTATE_*[n−1:0] The power state to which a transition is requested.

PACCEPT_* Indicates acknowledgment of the power state transition and completion of the power state transition in the device.

PDENY_* Indicates denial of the power state transition.

PACTIVE_* A hint signal that indicates opportunistic power state transitions such as dynamic retention modes. The DMC implementation defines the signal name and state transition hint.

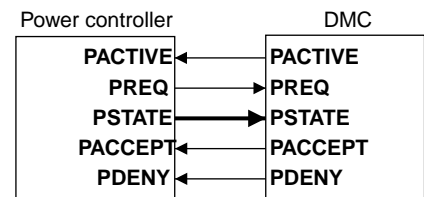The following figure shows the signals and their connections.

Figure 2-3 P-Channel interface with ACTIVE hint

P-Channel protocol

P-Channel protocol rules control P-Channel state transition.

The P-Channel protocol is as follows:

- **PREQ** can only transition from LOW to HIGH when **PACCEPT** and **PDENY** are both LOW.
- **PREQ** can only transition from HIGH to LOW when either:
  — **PACCEPT** is HIGH and **PDENY** is LOW.
  — **PACCEPT** is LOW and **PDENY** is HIGH.
- **PSTATE** can only transition when **PREQ**, **PACCEPT**, and **PDENY** are LOW. The signal transition must be guaranteed to be complete, and metastability resolved, when **PREQ** is asserted or **RESETn** is deasserted.
- **PACCEPT** can only transition from LOW to HIGH when **PREQ** is HIGH and **PDENY** is LOW.
- **PACCEPT** can only transition from HIGH to LOW when **PREQ** is LOW and **PDENY** is LOW.

- **PDENY** can only transition from LOW to HIGH when **PREQ** is HIGH and **PACCEPT** is LOW.
- **PDENY** can only transition from HIGH to LOW when **PREQ** is LOW and **PACCEPT** is LOW.

P-Channel state transition

This section describes the 4-phase handshake of the P-Channel.

The following figure shows a basic state transition timing diagram.
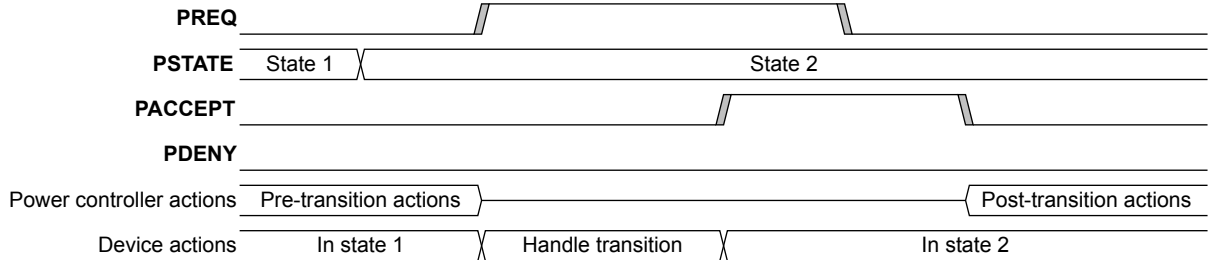

Figure 2-4 State transition timing diagram

The state transition uses the following 4-phase handshake:

1. The power controller drives the required state on **PSTATE**.
2. When it is guaranteed that this signal is stable, the power controller asserts **PREQ**.
3. The device asserts **PACCEPT**. If the state transition requires any actions from the device, such as cache initialization, the device must complete the action before it asserts **PACCEPT**.
4. The power controller responds by deasserting **PREQ**, and the device finishes by deasserting **PACCEPT**.

P-Channel on device reset

This section shows how to initialize the power state of a power domain.

The following figure shows the state initialization on reset. Certain device power states might power down the control logic. When powering this control logic back on, the power controller must indicate the state that the device must power up. The device detects the required state by sampling **PSTATE** when **RESETn** deasserts. The **PSTATE** inputs must be asserted before the deassertion of reset and remain after the deassertion of **RESETn**, to allow reset propagation within DMC. The power controller must ensure that the reset sequence is complete before transitioning **PSTATE**, otherwise the device might sample an undetermined value.
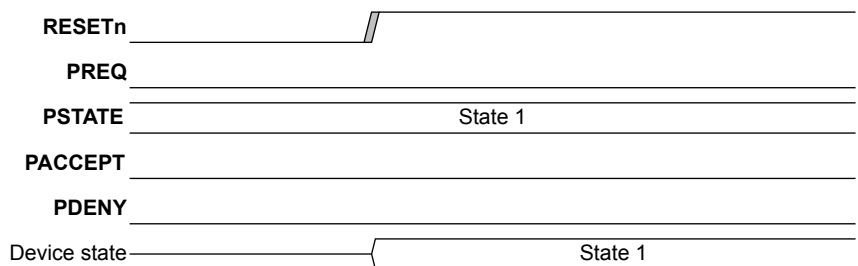

Figure 2-5 Reset state initialization

P-Channel interfaces

This section describes the various P-Channel interfaces in the DMC.

The following P-Channel interfaces manage the DMC power states:

Transitions to and from shutdown states

There are two types of shutdown state transitions:

Transitions to a shutdown state
  • ON to OFF.

Transitions from a shutdown state
  • OFF to ON

When the DMC transfers to the shutdown state, its control logic issues a **PDENY** to a **PREQ** on the LOGIC P-Channel if a PHY power down request has been requested through the arch_ph_lp register setting, and also if the PHY denies the power-down request from the DMC. The DMC does not deny a power-down request when there is ongoing system traffic, but waits for system traffic to subside before transferring to the OFF state.

PSTATE enumerations

The following table describes the PSTATE enumerations.

State	PSTATE value	PACTIVE value
ON	0x8	0x0100
OFF	0x0	0x0000

PSTATE on reset

The DMC enables entry into any pstate after reset. Set this state through the tie-off value of **pstate_init**.

──────── **Note** ────────

If you tie **pstate_init** to the OFF state, the DMC requires a P-channel request to move it to the ON state before any initial configuration process.

────────────────────

All **PSTATE** signals must be asserted at the deassertion of reset and all **PREQ** signals must be LOW at the deassertion of reset. Any **PSTATE** values other than those described here are invalid and can result in unpredictable behavior.

2.4 Constraints and limitations of use

The constraints and limitations of the DMC-500 depend on the SDRAMS used, and the interoperability within the PHYs. This, in turn, depends on the DFI parameters.

———— **Note** ————

The SDRAM device function is described in the JEDEC specifications that are global standards for the microelectronics industry.

————————————

2.5 System address conversion

The DMC-500 uses several function blocks to transform the system address into the SDRAM address. Read this to see the block descriptions and also to see under what circumstances the DMC rejects a transaction.

The following figure shows the functions that the DMC uses to transform the address that it receives from the system to the address it presents to the SDRAM.
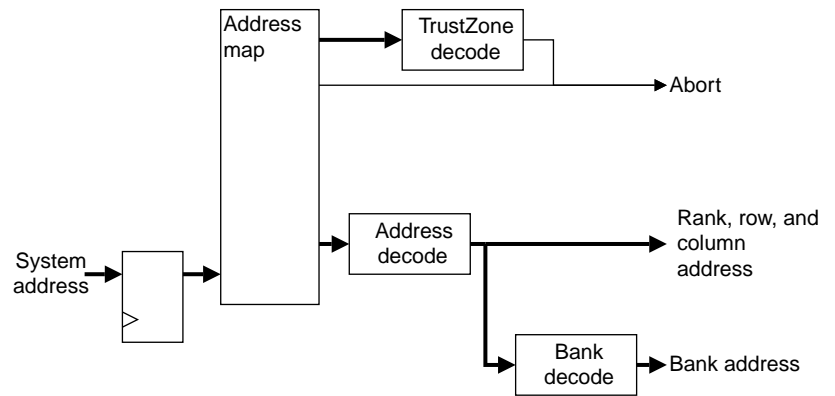

Figure 2-6 System address conversion

This section contains the following subsections:
- *2.5.1 Address map* on page 2-29.
- *2.5.2 TrustZone decode* on page 2-29.
- *2.5.3 Address decode* on page 2-29.
- *2.5.4 Bank decode* on page 2-29.

2.5.1 Address map

Receives the system address and converts it to a suitable form for the Address decode function.

2.5.2 TrustZone decode

Decodes invalid address regions, based on the values programmed into the `tz_region` registers.

TrustZone decode works on the pre-mapped full system address.

2.5.3 Address decode

Translates its input address to row, rank, bank, and column addresses.

2.5.4 Bank decode

Decodes the bank address using a function generated from the row address.

This is intended to increase available bandwidth by attempting to distribute traffic streams more efficiently across multiple banks. How effective it is is a function of the the incoming system traffic and decode options.

Appendix A
Revisions

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

It contains the following sections:

A.1 Revisions

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

Table A-1 Revision 0000-00

Change	Location	Affects
First release	-	-

Table A-2 Differences between issue 0000-00 and issue 0000-01

Change	Location	Affects
Updated example system diagram.	*1.1 About the product* on page 1-11	All revisions
Updated DMC functional block diagram.	*2.1 About the functions* on page 2-21	All revisions
Reset procedure updated	*2.2.2 Resets* on page 2-23	All revisions
Transitions from a shutdown state clarified.	*Transitions to and from shutdown states* on page 2-26	All revisions

Table A-3 Differences between issue 0000-01 and issue 0000-02

Change	Location	Affects
No technical changes.	-	-