MultiPort Memory Controller (GX175)

Revision: r0p2

Technical Reference Manual

ARM DDI 0277F

MultiPort Memory Controller (GX175)
Technical Reference Manual

Release Information

The following changes have been made to this document.

Change history

Date	Issue	Confidentiality	Change
12 March 2003	A	Non-Confidential	First release.
21 May 2003	B	Non-Confidential	Second release for r0p0. Various technical changes.
22 May 2003	C	Non-Confidential	Third release for r0p0. Reset values changed in MPMCDynamicReadConfig Register.
11 May 2004	D	Non-Confidential	Fourth release for r0p0. MPMCConfig register updated. Table 3-22 corrected.
5 October 2005	E	Non-Confidential	Update to r0p1. Revision field in Table 3-41 updated. Table 3-11 and Table 3-23 updated for t_{RAS} > RAS requirement.
23 November 2005	F	Non-Confidential	Update to r0p2.

Proprietary Notice

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
MultiPort Memory Controller (GX175) Technical Reference Manual

List of Tables
MultiPort Memory Controller (GX175) Technical Reference Manual

List of Figures
MultiPort Memory Controller (GX175) Technical Reference Manual

Preface

This preface introduces the *ARM MultiPort Memory Controller (GX175) Technical Reference Manual*. It contains the following sections:

About this document

This is the *Technical Reference Manual* (TRM) for the *ARM PrimeCell (GX175) MultiPort Memory Controller* (MPMC).

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, 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 manual is written for system designers, system integrators, and verification engineers who are designing a *System-on-Chip* (SoC) device that uses the MPMC. The manual describes the external functionality of the MPMC.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for a high-level view of the MPMC and a description of its features.

Chapter 2 *Functional Overview*

Read this chapter for a description of the major components of the MPMC and how they operate.

Chapter 3 *Programmer's Model*

Read this chapter for description of the MPMC registers.

Chapter 4 *Programmer's Model for Test*

Read this chapter for description of the MPMC test registers.

Appendix A *Signal Descriptions*

Read this appendix for a description of the MPMC input and output signals.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams*
- *Signals* on page xvi
- *Numbering* on page xvii.

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.

`mono`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.

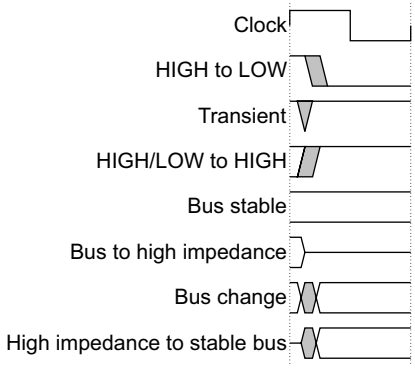
< and > Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example:

- `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
- The Opcode_2 value selects which register is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* on page xvi 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 and LOW for active-LOW signals.

Lower-case n Denotes an active-LOW signal.

Prefix A Denotes global *Advanced eXtensible Interface* (AXI) signals:

Prefix AR Denotes AXI read address channel signals.

Prefix AW Denotes AXI write address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix P Denotes *Advanced Peripheral Bus* (APB) signals.

Prefix R Denotes AXI read data channel signals.

Prefix W Denotes AXI write data channel signals.

Numbering

The numbering convention is:

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM Limited.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the ARM Frequently Asked Questions list.

ARM publications

This manual contains information that is specific to the MPMC. See the following documents for other relevant information:

- *AMBA Specification (Rev 2.0)* (ARM IHI 0011)
- *AMBA Design Kit Technical Reference Manual* (ARM DDI 0243)
- *MBX HR-S 3D Graphics Core Technical Reference Manual* (ARM DDI 0241)
- *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual* (ARM DDI 0249).

Feedback

ARM Limited welcomes feedback on the MPMC and its documentation.

Feedback on this product

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

- the product name
- a concise explanation of your comments.

Feedback on this manual

If you have any comments on this manual, send email to `errata@arm.com` giving:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the ARM MPMC (GX175). It contains the following sections:

1.1 About the ARM MPMC (GX175)

The MPMC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM Limited. It connects to the *Advanced High-performance Bus* (AHB) and a single dedicated MBX memory interface.

1.1.1 Features of the MPMC

The MPMC offers:

- AMBA 32-bit AHB compliance.
- Dedicated MBX Interface Port for direct connection to the ARM range of MBX 3D Graphics Cores.
- Dynamic memory interface supports SDRAM, DDR-SDRAM, and low-power variants. It also supports Micron SyncFlash types of memory.
- Asynchronous static memory device support including RAM, ROM, and Flash, with or without asynchronous page mode.
- Specifically designed for cached processors.
- Designed to work with noncritical word first, and critical word first processors, such as the ARM926EJ-S.
- Read and write buffers to reduce latency and to improve performance.
- Five AHB interfaces for accessing external memory.
- 8-bit, 16-bit, and 32-bit wide static memory support.
- 16-bit and 32-bit wide data bus SDRAM and SyncFlash memory support. 16-bit wide data bus DDR-SDRAM support.
- Static memory features include:
  - asynchronous page mode read
  - programmable wait states
  - bus turnaround cycles
  - output enable, and write enable delays
  - extended wait.
- Four chip selects for dynamic memory and four chip selects for static memory devices.

- Power-saving modes dynamically control **MPMCCKEOUT** and **MPMCCLKOUT**.
- Dynamic memory self-refresh mode supported by a *Power Management Unit* (PMU) interface or by software.
- Controller supports 2K, 4K, and 8K row address synchronous memory parts. That is, typical 512Mb, 256Mb, 128Mb, and 16Mb parts, with 8, 16, or 32 DQ (data) bits per device.
- Two reset domains enable dynamic memory contents to be preserved over a soft reset.
- A separate AHB interface for programming the MPMC registers. Enables the MPMC registers to be situated in memory with other system peripheral registers.
- Locked AHB transactions supported.
- Support for all AHB burst types.
- Little and big-endian support.
- Support for the *External Bus Interface* (EBI) that enables the memory controller pads to be shared.
- Integrated *Test Interface Controller* (TIC).
- PrimeCell ID support.

Note

Synchronous static memory devices (burst mode devices) are not supported.

1.2 Supported dynamic memory devices

This section provides examples of dynamic memory devices that are supported by the MPMC:

- *Examples of JEDEC DDR SDRAM devices*
- *Examples of JEDEC SDRAM devices*
- *Examples of Micron style SyncFlash devices* on page 1-5
- *Examples of JEDEC low-power SDRAM devices* on page 1-5.

Note

This is not an exhaustive list of supported devices.

1.2.1 Examples of JEDEC DDR SDRAM devices

The following JEDEC DDR-SDRAM devices are supported:

- 64Mb devices:
  - Micron MT46V2M32.
- 128Mb devices:
  - Micron MT46V16M8
  - Micron MT46V8M16.
- 256Mb devices:
  - Micron MT46V32M8
  - Micron MT46V16M16.

1.2.2 Examples of JEDEC SDRAM devices

The following JEDEC SDRAM devices are supported:

- 16Mb devices:
  - Micron MT48LC1M16A1S
  - Samsung K4S160822D-G/F
  - Samsung K4S641632C.
- 64Mb devices:
  - Micron MT48LC2M32B2-6
  - Micron MT28S4M162C-10
  - Elpida VDP4564323-10
  - Hitachi HM5264805F-75.

- 128Mb devices:
  - Micron MT48LC4M32B2
  - Micron MT48LC16M8A2
  - Micron MT48LC8M16A2.
- 256Mb devices:
  - Elpida VPP45256163-10
  - Micron MT48LC16M16A2-8E
  - Hitachi HM522532F-B6
  - Hitachi HM5225805B-75.
- 512Mb device:
  - Elpida HM5257805B-A6.

1.2.3 Examples of Micron style SyncFlash devices

The following 64Mb devices are supported:

- Micron MT28S4M16LC-10
- Micron MT28S4M16LC-12.

1.2.4 Examples of JEDEC low-power SDRAM devices

The following JEDEC low-power SDRAM devices are supported:

- 64Mb Micron MT48LC2M32LFFC-8
- 128Mb Infineon HYB25L128160AC.

1.3 Supported static memory devices

This section provides examples of static memory devices that are supported by the MPMC:

- *Examples of ROM devices*
- *Examples of page mode ROM devices*
- *Examples of SRAM devices*
- *Examples of flash devices*
- *Examples of page mode flash devices*.

Note

This is not an exhaustive list of supported devices.

1.3.1 Examples of ROM devices

The MPMC supports the 128Mb Samsung K3N9V100M-YC.

1.3.2 Examples of page mode ROM devices

The MPMC supports the 128Mb Samsung K3P9V100M-YC.

1.3.3 Examples of SRAM devices

The MPMC supports the following devices:

- 256Kb IDT IDT71V256SA20Y
- 256Kb Micron MT28F004b5-672
- 1Mb Micron MTSC2568-12
- 4Mb Samsung K6F8016R6M
- 4Mb Samsung K6R4016CK-12
- 8Mb Samsung K6T8016C3M-70
- 8Mb Samsung K6F8008R2M.

1.3.4 Examples of flash devices

The MPMC supports the 4Mb Micron MT28F004B5.

1.3.5 Examples of page mode flash devices

The MPMC supports the following devices:

- 8Mb Intel 28F800F3

- 4Mb Intel E28F320J3A110
- 8Mb AMD Am29BL802C.

1.4 Product revisions

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

r0p0-r0p1 Contains the following differences in functionality:

- MPMCPeriphID2 Register bits [7:4] now read back as `0x1`.
- There is no change to the functionality described in this manual. See the engineering errata that accompanies the product deliverables for more information.

r0p1-r0p2 Contains the following differences in functionality:

- MPMCPeriphID2 Register bits [7:4] now read back as `0x2`.
- There is no change to the functionality described in this manual. See the engineering errata that accompanies the product deliverables for more information.

Chapter 2
Functional Overview

This chapter describes the major functional blocks of the ARM MPMC (GX175). It contains the following sections:

2.1 MPMC functional description

The multiport memory controller block optimizes and controls external memory transactions. Figure 2-1 shows a block diagram of the MPMC.

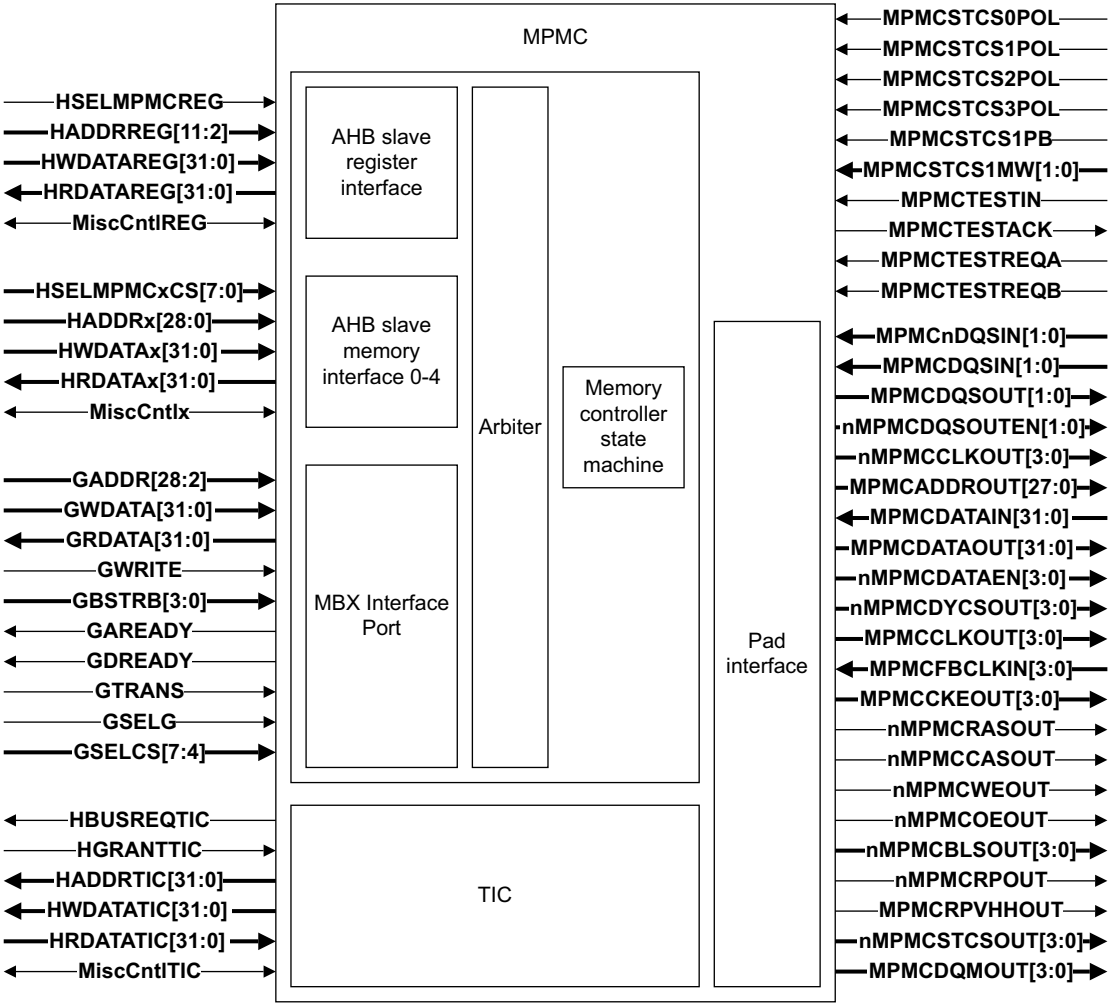


Figure 2-1 MPMC block diagram

Note

In Figure 2-1, the letter x used in the AHB slave memory interface signals denotes a number between 0 and 4.

The functions of the MPMC blocks are described in the following sections:

- *AHB slave register interface*
- *AHB slave memory interfaces* on page 2-4
- *MBX Interface Port* on page 2-5
- *Data buffers* on page 2-15
- *Arbiter* on page 2-27
- *Memory controller state machine* on page 2-27
- *Pad interface* on page 2-28
- *Test Interface Controller (TIC)* on page 2-29.

Note

For 32-bit wide chip selects, data is transferred to and from dynamic memory in single SDRAM bursts. For 16-bit wide chip selects, SDRAM bursts of two are used.

2.1.1 AHB slave register interface

The AHB slave register interface block enables the registers of the MPMC to be programmed. This module also contains most of the registers and performs the majority of the register address decoding. This interface must be connected to the ARM processor AHB bus to enable the MPMC to be programmed.

Memory transaction endianness and transfer width

To eliminate the possibility of endianness problems, all data transfers to and from the registers of the MPMC must be 32 bits wide.

Note

If an access is attempted with a size other than a word (32 bits), it causes an ERROR response on **HRESP** and the transfer is terminated.

2.1.2 AHB slave memory interfaces

The AHB slave memory interfaces enable devices to access the external memories. The memory interfaces are prioritized, with interface 0 having the highest priority. Having more than one memory interface gives high-bandwidth peripherals direct access to the MPMC, without data having to pass over the main system bus.

Note

- All AHB burst types are supported to enable the most efficient use of memory bandwidth.
- The AHB interfaces do not generate SPLIT and RETRY responses.

Memory transaction endianness

The endianness of the data transfers to and from the external memories is determined by the Endian mode (N) bit in the MPMCConfig register.

Note

The memory controller must be idle, see *Status Register, MPMCStatus* on page 3-11, before endianness is changed, so that the data is transferred correctly.

Memory transaction size

Memory transactions can be 8, 16, or 32 bits wide. Any access attempted with a size greater than a word (32 bits) causes an ERROR response on **HRESP** and the transfer is terminated.

Write-protected memory areas

Write transactions to write-protected memory areas generate an ERROR response on **HRESP** and the transfer is terminated.

2.1.3 MBX Interface Port

The MBX Interface Port is described in:

- *Slave select signal generation*
- *Transaction descriptions*
- *Typical MBX 3D Graphics Core data transfer profiles* on page 2-14.

Slave select signal generation

The MBX interface port can only access dynamic memory devices using input signals **GSELCS[7:4]**. Static memory cannot be accessed by the MBX interface port. The **GSELG** and **GSELCS[7:4]** signals must be generated by a decoder between the MBX 3D Graphics Core and the MPMC. This is similar to an AHB decoder, where the upper bits of the current address are decoded to select the MPMC and the chip select to use for the memory transfer.

If there are multiple slaves connected to the MBX 3D Graphics Core then the above decoding is required for the **GSELG** signal. However, if the MPMC is the only slave then the **GTRANS** signal can be used instead because every valid transfer selects the MPMC.

If the MBX 3D Graphics Core accesses memory on more than one chip select then the above decoding is required for the **GSELCS[7:4]** signals. However, if only one chip select is used then this bit of **GSELCS** can be driven in the same manner as **GSELG**, with the other **GSELCS** bits tied LOW.

Transaction descriptions

Data is transferred on any cycle when **GTRANS** and **GAREADY** are both HIGH on a rising **HCLK** edge. MBX 3D Graphics Core transactions are described in the following sections:

- *Read transactions* on page 2-6
- *Write transactions* on page 2-9
- *Read/write transactions* on page 2-12
- *GBSTRB functionality* on page 2-12
- *Additional considerations* on page 2-13.

Note

The MBX only supports little-endian format transfers, and the operation of the MBX Interface Port is unaffected by the **MPMCBIGENDIAN** input or N bit of the MPMCConfig Register. Reads of data written by the MBX are always performed correctly.

However, the byte ordering of data can be changed in the following situations when the MPMC is configured to be in big-endian mode:

- the MBX reads data written by an AHB device in big-endian format
- an AHB device reads data written by the MBX in little-endian format.

Read transactions

Read transfers are split into two distinct phases

- address phase
- data phase.

To start a read transfer, the MBX 3D Graphics Core memory interface:

- drives **GWRITE** LOW
- drives an address onto **GADDR**
- drives **GBSTRB**
- indicates that these signals are valid by driving **GTRANS** HIGH.

This effectively requests use of the memory, and the transfer is initiated when the MPMC drives **GAREADY** HIGH if it is not already HIGH. Read transfers then occur on every subsequent clock, unless **GAREADY** is taken LOW, effectively inserting wait states into the address phase.

Data is returned, in order, by the MPMC at a later time on the **GRDATA** bus. The data on this bus is valid when **GDREADY** is HIGH. This is a split protocol. The ordering of **GRDATA** responses is the same as **GADDR** responses, because there is no out-of-order return of data.

Figure 2-2 on page 2-7 shows a read of five words by the MBX 3D Graphics Core. The diagram shows the memory access as having a latency of eight clocks.

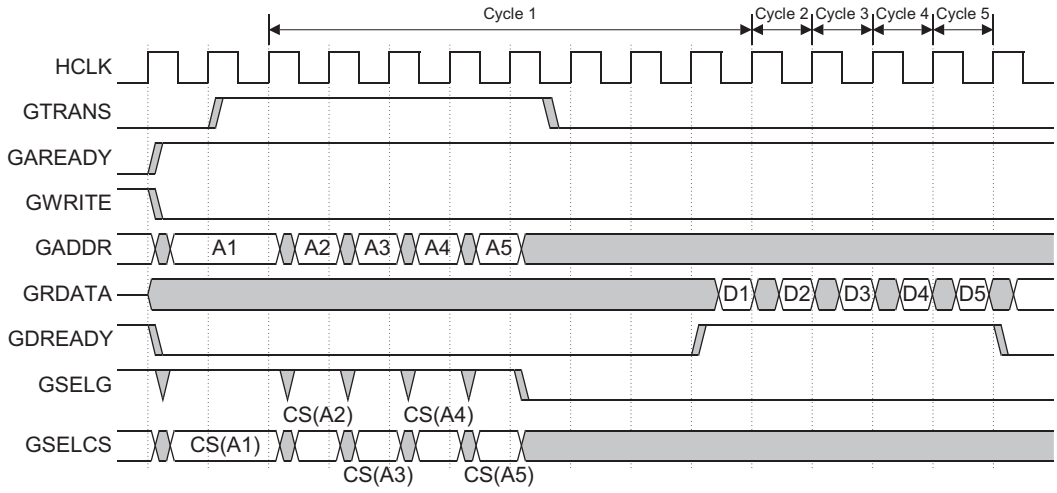


Figure 2-2 MBX 3D Graphics Core read, five transactions

Figure 2-3 on page 2-8 shows a large number of read transactions. The last three transactions are held off for two clocks by the MPMC by driving **GAREADY** LOW, for example, because of buffers in the MPMC filling. It also shows returned data being marked as invalid, with **GDREADY** being driven LOW. This adds more latency to the held-off transactions. This might be caused by the MPMC having to open a new memory page to access address A7 and return D7.

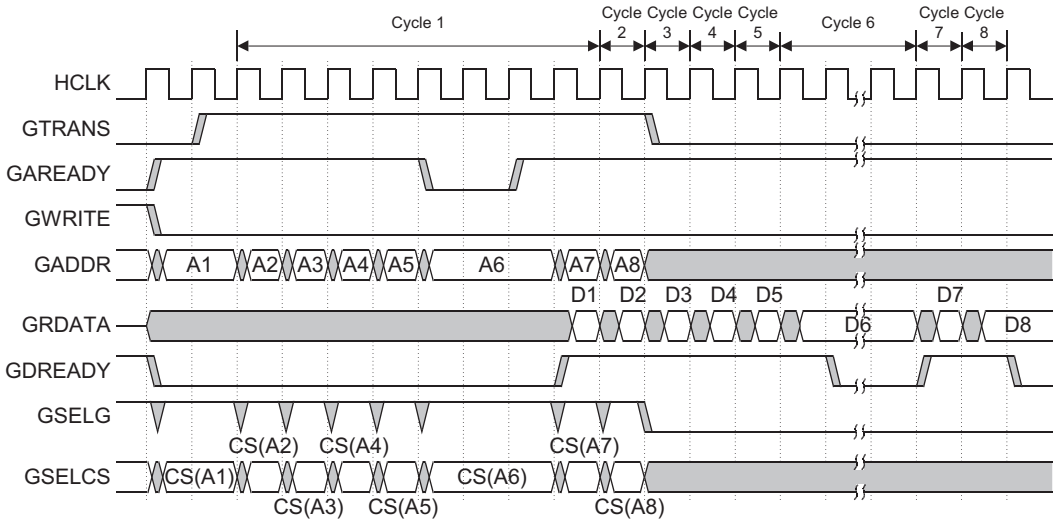


Figure 2-3 MBX 3D Graphics Core read, multiple transactions

Figure 2-4 shows the MBX 3D Graphics Core pausing its activity on the memory interface by driving **GTRANS** LOW. Addresses A4, A5, and A6 are not read.

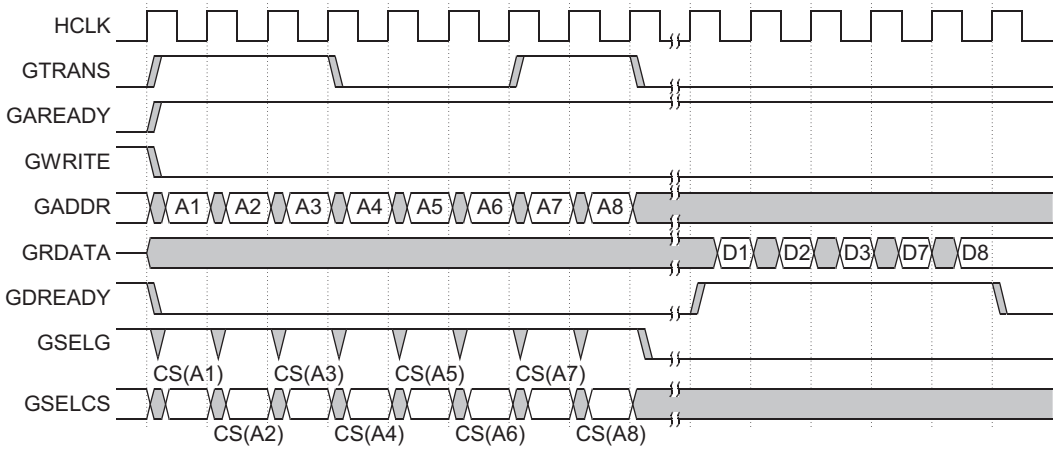


Figure 2-4 MBX 3D Graphics Core read, paused

Figure 2-5 shows a series of read request transactions on the MBX 3D Graphics Core memory interface. The address requests consist of five individual requests:

- the first is performed without delay
- the second is stalled by the MBX 3D Graphics Core for one clock cycle
- the third access is stalled by both the MPMC and the MBX 3D Graphics Core for three clock cycles because of the combined effects of the **GAREADY** and **GDREADY** signals
- the MPMC returns the read data with the fourth and fifth pieces of data being stalled for one clock cycle.

This also demonstrates that return data can happen concurrently with address request cycles.

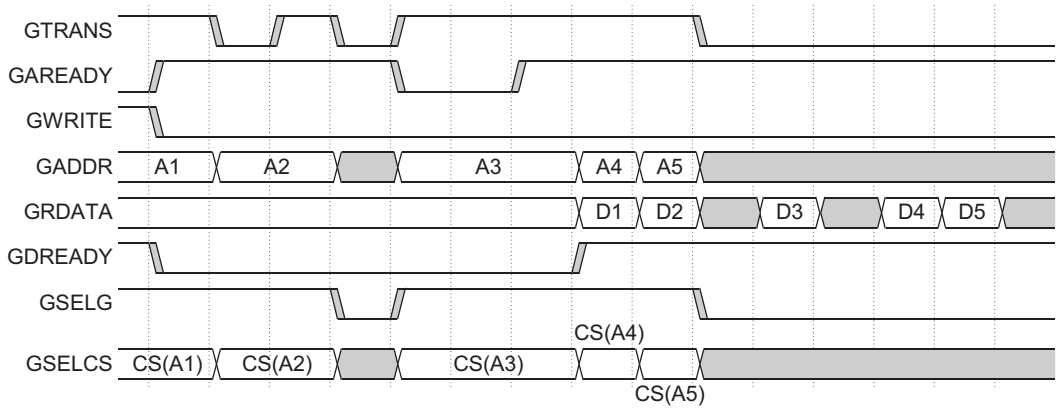


Figure 2-5 MBX 3D Graphics Core read, stalled

Write transactions

To start a write transfer, the MBX 3D Graphics Core memory interface:

- drives **GWRITE** HIGH
- drives **GADDR**, **GWDATA**, and **GBSTRB**
- signals that these signals are valid by driving **GTRANS** HIGH.

This effectively requests use of the memory, and the transfer is initiated when the MPMC drives **GAREADY** HIGH, if it is not already. Write transfers then occur on every subsequent clock, unless **GAREADY** is taken LOW, inserting wait states into the transfer.

Note

The value of **GDREADY** is ignored on writes.

Figure 2-6 shows a series of eight write transactions.

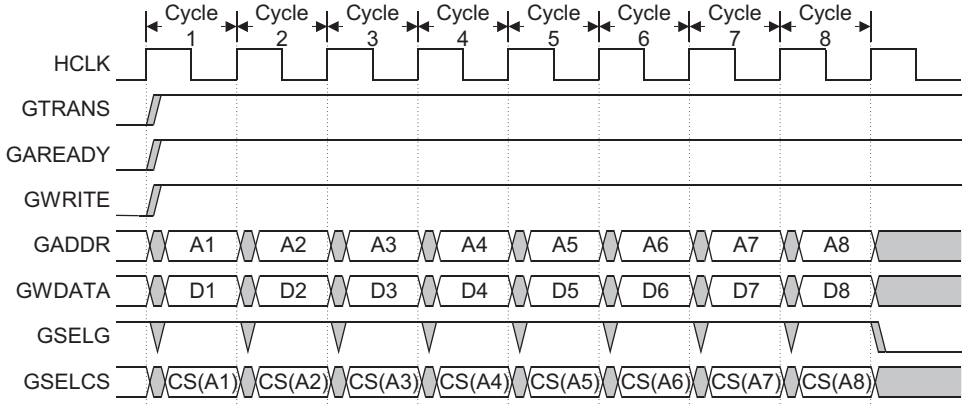


Figure 2-6 MBX 3D Graphics Core write, eight transactions

Figure 2-7 on page 2-11 shows five write cycles. The first and fifth cycles are extended by the memory controller, by driving **GAREADY** LOW.

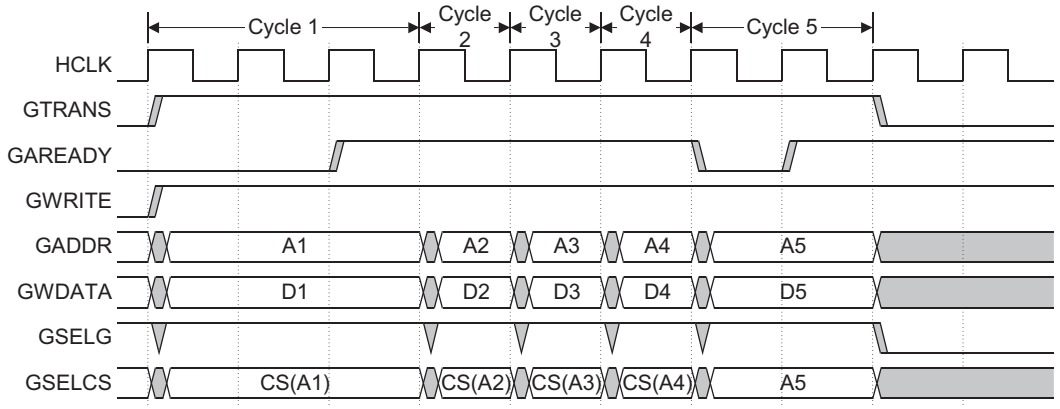


Figure 2-7 MBX 3D Graphics Core write, five transactions

Figure 2-8 shows six write cycles. **GTRANS** LOW indicates that the data on the address and data buses, **GADDR** and **GWDATA** respectively, are not valid and so no write transactions take place for those clock periods despite **GAREADY** being HIGH.

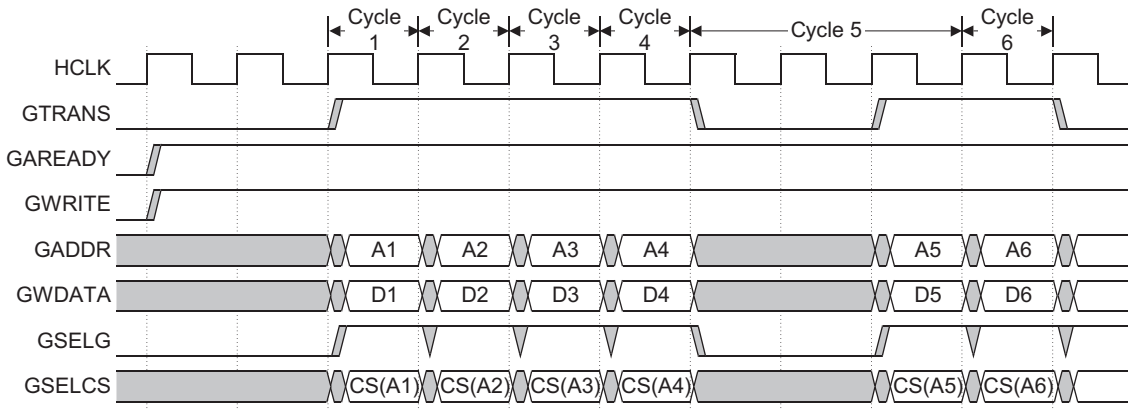


Figure 2-8 MBX 3D Graphics Core write, six transactions

Read/write transactions

Figure 2-9 shows a series of write request transactions on the MBX 3D Graphics Core memory interface with read request data being returned from previous read request transactions. This shows that read data can overlap with write requests, ensuring maximum bus efficiency.

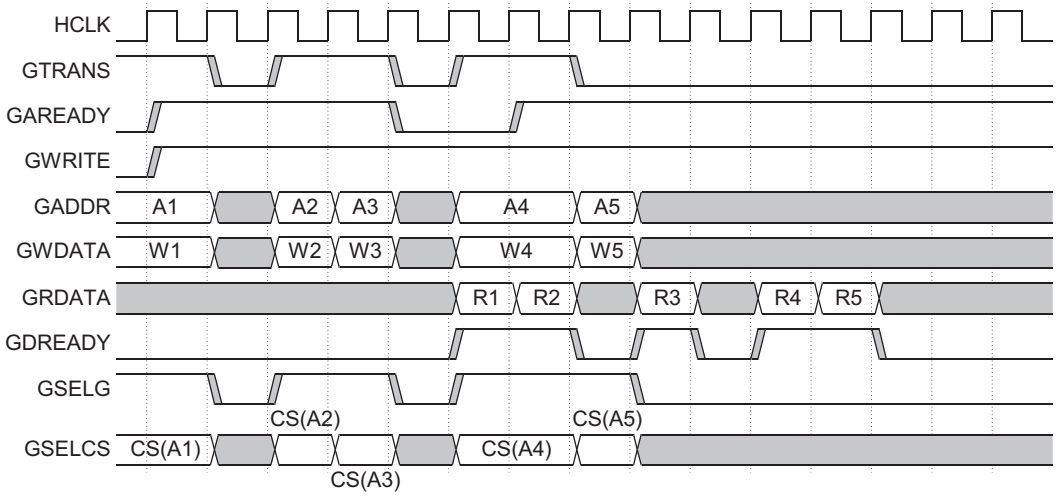


Figure 2-9 MBX 3D Graphics Core read/write transactions

GBSTRB functionality

The MBX interface uses the **GBSTRB** as byte lane strobes to enable subword writes.

GBSTRB is a four-bit byte lane enable signal. A HIGH on the relevant byte mask bit indicates that the byte is valid and must be written to SDRAM.

Table 2-1 lists which bits of **GBSTRB** relate to which byte lanes in **GWDATA**.

Table 2-1 GBSTRB bits

GBSTRB bit	GWDATA byte
[0]	[7:0]
[1]	[15:8]
[2]	[23:16]
[3]	[31:24]

Figure 2-10 shows how **GBSTRB** functions. It shows six write transactions to SDRAM through the memory controller.

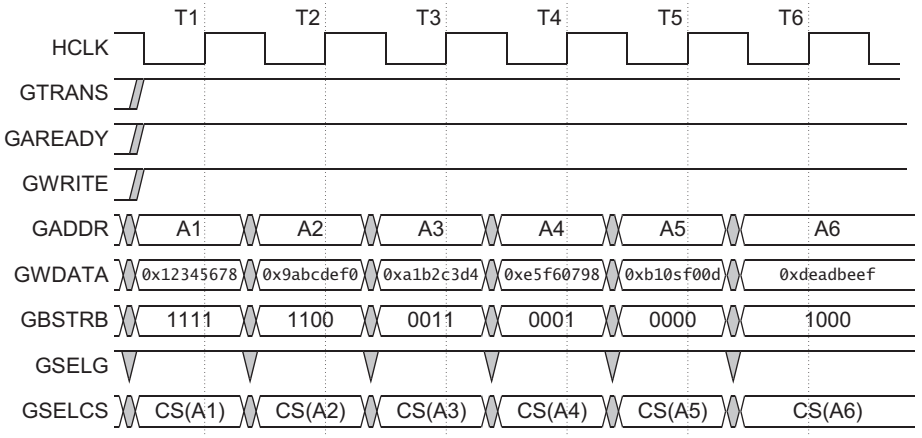


Figure 2-10 GBSTRB timing diagram

Table 2-2 lists the GBSTRB transactions.

Table 2-2 GBSTRB transactions

Transaction	Activity
T1	Writes the complete 32-bit data word to address A1.
T2	Writes `0x9abc` to the top 16 bits of address A2. The bottom 16 bits remain unchanged.
T3	Writes `0xc3d4` to the bottom two bytes of address A3. The top16 bits remain unchanged.
T4	Writes `0x98` to the bottom byte of address A4.
T5	Does not change the data word currently contained at address A5.
T6	Writes `0xde` to the top byte (bits 31:24) of address A6.

Additional considerations

The MPMC only initiates a transaction when both **GTRANS** and **GAREADY** are HIGH.

Typical MBX 3D Graphics Core data transfer profiles

Memory accesses originating from the MBX 3D Graphics Core originate from a variety of internal sources. Table 2-3 lists an example relative distribution of these accesses for the ARM MBX HR-S.

Note

Burst does not mean that the accesses are to contiguous addresses as is the case with AMBA. It implies that there is a group of requests originating from a given internal port, that are generally in the same memory page.

Table 2-3 Example ARM MBX HR-S memory transfer profile

Access port	Burst type	Read/ write	Access type	Scalable	Bandwidth (MB/s)	% of bandwidth
TARW	Usually burst	RW	Random (usually in page)	Yes	17.01	18.28
PARAM (ISP pointer read)	Bursts of 4	Read	Random (usually in page)	Yes	2.23	2.40
PARAM (ISP vertex read)	Bursts of 12	Read	Random (usually in page)	Yes	27.3	29.33
PARAM (TSP vertex read)	Bursts of 20	Read	Random (usually in page)	Yes	26.38	28.34
ZLZS (Z load/Z store port)	256 word bursts	RW	Random (usually in page)	Yes	0	0
TEX (texture cache port)	Bursts of 2 or 4	Read	Random (often not in page)	Yes	15.55	16.71
PIX (pixel write port)	256 word bursts	Write	Contiguous in page	No	4.61	4.95
Total bandwidth					93.08	100

Note

Because Z load/store is not normally enabled in this typical example case, Z load/store bandwidth is zero. The proportion of the bandwidth that each port requires depends on the input data. The figures in Table 2-3 on page 2-14 represent a profile for a system based on the following data:

- MBX HR-S core
- screen size of 320x240 pixels
- screen color depth of 16bpp
- frame rate of 30 *frames per second* (fps)
- input number of vertices is 20000
- texture mode of 16bpp
- texture cache hit rate of 85%.

The MBX core arbitrates internally between the requestors.

The MBX Interface Port is the lowest priority port on the MPMC. MBX 3D Graphics Core accesses are high-bandwidth but are not latency-critical. MBX 3D Graphics Core are therefore only allocated what memory bandwidth is left over after all other devices have used what they require. This ensures that the MBX Interface Port cannot starve other devices of bandwidth.

2.1.4 Data buffers

Each AHB port contains a single 32-bit word combined read and write buffer. These buffers are used to maximize memory performance for 8-bit wide or 16-bit wide read or write multi-transfer AHB bursts. Each AHB port buffer can be enabled or disabled using the MPMCAHBControl Register, Buffer enable (E) field.

Note

The MBX interface port does not contain a buffer similar to the AHB ports. All transfers through the MBX interface port are 32-bit with optional byte lane strobes and do not form parts of bursts, so a buffer is not required to maximize memory performance for bursts with a size of less than 32-bit.

Read transaction buffer operation

If an 8-bit or 16-bit wide AHB read transfer is performed with the buffers enabled, the read transaction submitted to the memory is at the maximum width of the memory chip-select. Therefore a chip-select with a 32-bit data bus returns 32-bits of data. This data is then placed into the buffer. Data is provided from the buffer to the AHB bus as necessary. This reduces the number of read transactions to external memory for 8-bit or 16-bit wide AHB transactions. The memory controller can then rearbitrate to a different AHB port and perform memory transactions while the data is being transferred from the data buffer.

Note

- Enabling the buffers has no impact on 32-bit wide read transactions.
- The memory controller rearbitrates to an open page transfer during a buffered transaction.

Enabling read buffer

For read transactions the respective AHB port buffer is only used if:

- The AHB respective AHB port buffer is enabled.
- Either 8-bit or 16-bit wide AHB read transactions are performed. 32-bit wide transactions do not make use of the buffer, as this does not improve performance.
- A multi-word AHB burst is performed. AHB burst SINGLE transfers do not use the buffer.
- AHB HPROT protection information indicates that the transfer is cacheable, indicating to the memory controller that the particular read transaction can be buffered.
- The transaction is not a locked transfer (**HMASTLOCK** is LOW). This ensures that atomic AHB transactions are performed straight to memory and are not rearbitrated during the transaction.

Note

If an AHB master does not provide AHB HPROT protection information the relevant **HPROT** signals can be tied-off as required.

Read data re-use

Only the AHB burst that fetched the data from the memory into the buffer can use the data in the buffer. Subsequent AHB bursts do not make use of the read data in the buffer even if the transaction is to the same memory area.

Advantages of read buffering

Enabling read buffering:

- reduces power consumption because fewer commands are issued to memory.
- increases memory bandwidth for 8-bit and 16-bit wide memory transactions, because the memory controller can rearbitrate to service a different AHB port while the data is being read from the buffer.

Note

Only AHB ports that have a memory request to open pages of SDRAM memory are serviced.

Read buffer example

This example describes the case where the AHB port buffers are enabled and cacheable transfers are performed, with AHB port 0 and AHB port 1 performing INCR4, 16-bit wide read transactions. Accesses are to a 32-bit wide dynamic memory chip-select.

Read buffer enabled

Table 2-4 shows that with the read buffer enabled the memory is operating at maximum efficiency and providing 32 bits of data on each clock cycle.

Table 2-4 Read buffer enabled

Clock cycle	AHB0	AHB1
1	32-bit read from memory and placed in buffer. Data 0 returned on AHB	AHB port waiting for data
2	Data 1 returned on AHB	32-bit read from memory and placed in buffer. Data 0 returned on AHB

Table 2-4 Read buffer enabled (continued)

Clock cycle	AHB0	AHB1
3	32-bit read from memory and placed in buffer. Data 2 returned on AHB	Data 1 returned on AHB
4	Data 3 returned on AHB	32-bit read from memory and placed in buffer. Data 2 returned on AHB
5	AHB port available	Data 3 returned on AHB

Read buffer disabled

Table 2-5 shows that with the read buffer disabled the memory provides 16 bits of data on each clock cycle. The memory controller therefore only provides half the amount of bandwidth compared to the case with the buffers enabled.

Table 2-5 Read buffer disabled

Clock cycle	AHB0	AHB1
1	16-bit read from memory. Data 0 returned on AHB	AHB port waiting for data
2	16-bit read from memory. Data 1 returned on AHB	AHB port waiting for data
3	16-bit read from memory. Data 2 returned on AHB	AHB port waiting for data
4	16-bit read from memory. Data 3 returned on AHB	AHB port waiting for data
5	AHB port available	16-bit read from memory. Data 0 returned on AHB
6	AHB port available	16-bit read from memory. Data 1 returned on AHB
7	AHB port available	16-bit read from memory. Data 2 returned on AHB
8	AHB port available	16-bit read from memory. Data 3 returned on AHB

Disadvantages of read buffering

The disadvantage with enabling read buffering is that a 8-bit or 16-bit wide read transfer can take longer to complete, as the AHB port is rearbitrated to maximize bandwidth while the data is being read from the buffer. If buffering is not desirable then read buffering can be disabled by either:

- setting the Buffer enable (E) field of the MPMCAHBControl register inactive
- setting the AHB HPROT protection information to indicate a nonbufferable access.

Worst case additional latency

The worst case additional latency is for the case where one AHB port, in this example AHB port 0 performs INCR16 16-bit wide read transactions. Other AHB ports, in this case AHB port 1 and 2, perform continuous, in page, INCR16 32-bit read transactions. Accesses are to a 32-bit wide dynamic memory chip-select.

Read buffer enabled

With the read buffer enabled the memory provides 32 bits of data on every clock cycle. The memory controller rearbitrates to a different AHB port when data is being read from the buffer. The latency of a buffer transfer might therefore be longer than a nonbuffered transfer.

The worst case additional latency for a buffered transfer is when a buffered INCR16 16-bit wide burst is being performed and the other AHB ports are performing INCR16 32-bit wide transfers. Table 2-6 shows that the INCR16 16-bit wide read can take up to 128 cycles to complete.

Table 2-6 Read buffer enabled

Clock cycle	AHB0	AHB1	AHB2
1	32-bit read from memory and placed in buffer. 16-bit data 0 returned on AHB	AHB port waiting for data	AHB port waiting for data
2	16-bit data 1 returned on AHB	AHB port waiting for data	AHB port waiting for data
3	AHB port waiting for data	32-bit read zero from memory	AHB port waiting for data
4-17	AHB port waiting for data	32-bit read 1-14 from memory	AHB port waiting for data
18	AHB port waiting for data	32-bit read 15 from memory	AHB port waiting for data

Table 2-6 Read buffer enabled (continued)

Clock cycle	AHB0	AHB1	AHB2
19	32-bit read from memory and placed in buffer. 16-bit data 2 returned on AHB	Next INCR16 transfer	AHB port waiting for data
20	16-bit data 3 returned on AHB	AHB port waiting for data	AHB port waiting for data
21	AHB port waiting for data	AHB port waiting for data	32-bit read 0 from memory
22-37	AHB port waiting for data	AHB port waiting for data	32-bit read 1-14 from memory
38	AHB port waiting for data	AHB port waiting for data	32-bit read 15 from memory
39	AHB port waiting for data	AHB port waiting for data	Next INCR16 transfer
40	32-bit read from memory and placed in buffer. 16-bit data 4 returned on AHB	AHB port waiting for data	AHB port waiting for data
41	16-bit data 5 returned on AHB	AHB port waiting for data	AHB port waiting for data

Note

Table 2-6 on page 2-19 lists the first 41 cycles of the transaction. Cycles 42-136 are a continuation of the sequence shown.

Read buffer disabled

Table 2-7 shows that if read buffers are not enabled, an INCR16 16-bit wide read burst takes 16 cycles to complete when the read data starts being returned from the memory

Table 2-7 Read buffer disabled

Clock cycle	AHB0	AHB1	AHB2
1	16-bit read from memory. Data 0 returned on AHB	AHB port waiting for data	AHB port waiting for data
2-15	16-bit read from memory. Data 1-14 returned on AHB	AHB port waiting for data	AHB port waiting for data
16	16-bit read from memory. Data 15 returned on AHB	AHB port waiting for data	AHB port waiting for data
17	AHB port available	32-bit read from memory. Data 0 returned on AHB	AHB port waiting for data
18-31	AHB port available	32-bit read from memory. Data 1-14 returned on AHB	AHB port waiting for data
32	AHB port available	32-bit read from memory. Data 15 returned on AHB	AHB port waiting for data

Write transaction buffer operation

If an 8-bit or 16-bit wide AHB write transfer is performed with the buffers enabled the write data is merged into the buffer, so that a write of the maximum width of the memory chip-select is performed. Therefore, for a chip-select with a 32-bit data bus, 32 bits of data are written at a time. This reduces the number of write transactions to external memory for 8-bit or 16-bit wide AHB transactions. The memory controller can then re-arbitrate to a different AHB port and perform memory transactions while the data is being written into the data buffer.

Note

- Enabling the buffers has no impact on 32-bit wide write transactions.
- The memory controller re-arbitrates to an open page transfer during a buffered transaction.

Enabling write buffer

For write transactions the respective AHB port buffer is only used if:

- The respective AHB port buffer is enabled.
- An 8-bit or 16-bit write transaction is performed. 32-bit wide transactions do not make use of the buffer because this does not improve performance.
- A multi-word AHB burst is performed. AHB burst SINGLE transfers do not use the buffer.
- AHB HPROT protection information indicates that the transfer is bufferable. This indicates to the memory controller that the particular write transaction can be buffered.
- The transaction is not a locked transfer (**HMASTLOCK** is LOW). This ensures that atomic AHB transactions are performed straight to memory and are not re-arbitrated during the transaction.

Note

If an AHB master does not provide AHB HPROT protection information, then you can tie off the relevant **HPROT** signals.

Write data re-use

If an AHB port performs a burst write into a buffer subsequent AHB burst writes do not merge data into the same buffer, even if the subsequent burst is to the same area of memory. Instead the data in the buffer from the first write is submitted to memory and only then can the second burst start. This ensures that the memory is updated at the end of each burst write so that if another AHB port reads from the same memory location the memory is updated.

Advantages of write buffering

Enabling write buffering:

- reduces power consumption as fewer commands are issued to memory
- increases memory bandwidth for 8-bit and 16-bit wide memory transactions, because the memory controller can re-arbitrate to service a different AHB port while the data is being written to the buffer.

Note

Only AHB ports that have a memory request to open pages of SDRAM memory are serviced.

Write buffer example

The write buffer example describes the case where the AHB port buffers are enabled and bufferable transfers are performed, with AHB port 0 and AHB port 1 performing INCR4, 16-bit wide write transactions. Accesses are to a 32-bit wide dynamic memory chip-select.

Write buffer enabled

Table 2-8 shows that with the write buffer enabled the memory is operating at maximum efficiency and providing 32 bits of data on each clock cycle.

Table 2-8 Write buffer enabled

Clock cycle	AHB 0	AHB 1
1	16-bit write data 0 written into buffer	AHB port waiting for data
2	16-bit write data 1 written into buffer. 32-bit write data written to memory	16-bit write data 0 written into buffer
3	16-bit write data 2 written into buffer	16-bit write data 1 written into buffer. 32-bit write data written to memory
4	16-bit write data 3 written into buffer	16-bit write data 2 written into buffer
5	AHB port available	16-bit write data 3 written into buffer. 32-bit write data written to memory

Write buffer disabled

Table 2-9 shows that with the write buffer disabled 16 bits of data is written to memory on each clock cycle. The memory controller therefore only provides half the amount of bandwidth compared to the case with the buffers enabled.

Table 2-9 Write buffer disabled

Clock cycle	AHB 0	AHB 1
1	16-bit write data 0 written into memory	AHB port waiting for data
2	16-bit write data 1 written into memory	AHB port waiting for data
3	16-bit write data 2 written into memory	AHB port waiting for data
4	16-bit write data 3 written into memory	AHB port waiting for data
5	AHB port available	16-bit write data 0 written into memory
6	AHB port available	16-bit write data 1 written into memory
7	AHB port available	16-bit write data 2 written into memory
8	AHB port available	16-bit write data 3 written into memory

Disadvantages of write buffering

The disadvantage with enabling write buffering is that an 8-bit or 16-bit wide write transfer can take longer to complete, because the AHB port is re-arbitrated to maximize bandwidth while the data is being written into the buffer. If buffering is not required, write buffering can be disabled by:

- setting the Buffer enable (E) field of the MPMCAHBControl Register inactive
- setting the AHB HPROT protection information to indicate a nonbufferable access.

Worst case additional latency

The worst case additional latency is for the case where one AHB port, in this case AHB port 0, performs INCR16 16-bit wide write transactions. Other AHB ports, in this case AHB port 1 and 2, perform continuous, in page, INCR16 32-bit write transactions. Accesses are to a 32-bit wide dynamic memory chip-select.

Write buffer enabled

With the write buffer enabled the memory writes 32 bits of data on every clock cycle.

The memory controller re-arbitrates to a different AHB port when data is being written to the buffer. The latency of a buffer transfer might therefore be longer than a nonbuffered transfer.

The worst case additional latency for a buffered transfer is when a buffered INCR16 16-bit wide burst is being performed and the other AHB ports are performing INCR16 32-bit wide transfers. See Table 2-10 for an example of the worse case scenario.

Table 2-10 Write buffer enabled

Clock cycle	AHB 0	AHB 1	AHB 2
1	16-bit write data 0 written into buffer	AHB port waiting for data	AHB port waiting for data
2	16-bit write data 1 written into buffer. 32-bit write to memory	AHB port waiting for data	AHB port waiting for data
3	16-bit write data 2 written into buffer	32-bit write 0 to memory	AHB port waiting for data
4	16-bit write data 3 written into buffer	32-bit write 1 to memory	AHB port waiting for data
5-17	AHB port waiting for data	32-bit write 2-14 to memory	AHB port waiting for data
18	AHB port waiting for data	32-bit write 15 to memory	AHB port waiting for data
19	64-bit write to memory	Next INCR16 transfer	AHB port waiting for data
20	16-bit write data 4 written into buffer	AHB port waiting for data	AHB port waiting for data
21	16-bit write data 5 written into buffer	AHB port waiting for data	32-bit write 0 to memory
23-37	AHB port waiting for data	AHB port waiting for data	32-bit write 1-14 to memory
38	AHB port waiting for data	AHB port waiting for data	32-bit write 15 to memory
39	AHB port waiting for data	AHB port waiting for data	Next INCR16 transfer
40	32-bit write to memory	AHB port waiting for data	AHB port waiting for data

Note

Table 2-10 lists the first 40 cycles of the transaction. Cycles 41 to 122 are a continuation of the sequence shown.

Write buffer disabled

Table 2-11 shows that if write buffers are not enabled an INCR16 16-bit wide write burst takes 16 cycles to complete when the write data starts being written to memory.

Table 2-11 Write buffer disabled

Clock cycle	AHB 0	AHB 1	AHB 2
1	16-bit write 0 to memory	AHB port waiting for data	AHB port waiting for data
2-15	16-bit write 1-14 to memory	AHB port waiting for data	AHB port waiting for data
16	16-bit write 15 to memory	AHB port waiting for data	AHB port waiting for data
17	AHB port available	32-bit write 0 to memory	AHB port waiting for data
18-31	AHB port available	32-bit write 1-14 to memory	AHB port waiting for data
32	AHB port available	32-bit write 15 to memory	AHB port waiting for data
33	AHB port available	AHB port available	32-bit write 0 to memory
34-47	AHB port available	AHB port available	32-bit write 1-14 to memory
48	AHB port available	AHB port available	32-bit write 15 to memory
49	AHB port available	AHB port available	AHB port available

Zero wait state write transfers

When the buffers are disabled, all write transfers receive **HREADY** LOW wait states until the transfer has started to be performed by the MPMC. **HREADY** then goes HIGH. This functionality is important for multi-port masters to ensure data coherency over multiple ports, or when data is being passed between masters. For example, the ARM11 read and write data ports require the write transfers to receive wait states until the write is being performed to ensure coherency between the two ports. This is the default operation of the MPMC after reset.

With the buffers enabled and empty, a write transfer receives no wait states and completes immediately, enabling reduced write latency if there are no data coherency issues. Subsequent writes are waited until the first write has completed.

To enable zero wait state writes to be performed without using the data merge buffers, the buffers must be enabled and the **HPROT[3:2]** lines driven correctly to indicate that reads are not cacheable and writes are not bufferable.

2.1.5 Arbiter

The arbiter arbitrates between the AHB slave memory interfaces. AHB port 0 has the highest access priority, and the MBX Interface Port has the lowest priority. For more information see *Arbitration* on page 2-37.

2.1.6 Memory controller state machine

The memory controller state machine comprises two functional blocks:

- a static memory controller
- a dynamic memory controller.

Low transaction latency and high memory bandwidth are conflicting design parameters.

A memory controller designed to support high bandwidth has logic to reorder transactions to maximize memory efficiency. This logic increases transaction latency.

A memory controller designed to reduce latency has less logic for the transaction to pass through, reducing the amount of logic used to maximize the transaction efficiency. This decreases the supported memory bandwidth.

The memory controllers have been designed with both these parameters in mind, and have both good latency and memory bandwidth.

Providing multiple AHB memory ports in the memory controller enables the memory bandwidth to be shared over multiple AHB buses. This reduces the load on performance-critical buses.

To make full use of dynamic memory bandwidth a multiple-port design is required so that:

- the transaction order can be rearranged to maximize the number of in page accesses
- the dynamic memory transactions can be pipelined
- the dynamic memory transactions can be interleaved.

2.1.7 Pad interface

The pad interface block provides the interface to the pads. The pad interface uses a feedback clock, **MPMCFBCLKIN[3:0]**, to capture read data for SDR-SDRAM memory devices. To capture read data for DDR-SDRAM devices, the data strobe signals, **MPMCDQSIN[1:0]** and **nMPMCDQSIN[1:0]**, are used together with an external *Delay Locked Loop* (DLL) block to resynchronize from the off-chip to on-chip domains.

Figure 2-11 shows a block diagram of the pad interface.

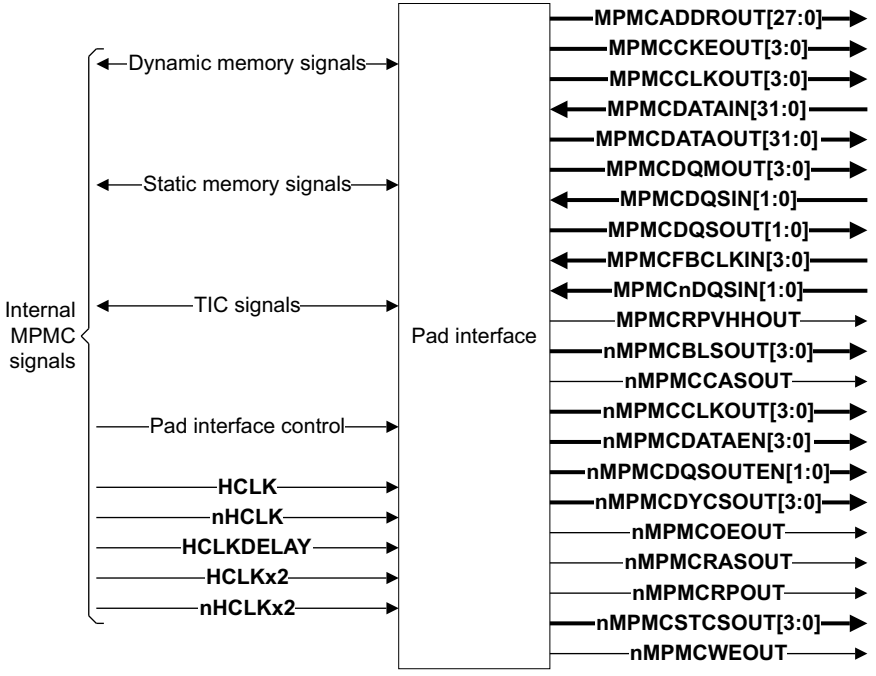


Figure 2-11 Pad interface block diagram

2.1.8 Test Interface Controller (TIC)

The TIC enables TIC device testing. For full details about the TIC, see the *AMBA Specification*. The AHB master interface must normally be placed on the same AHB interface as the ARM processor. Figure 2-12 shows a block diagram of the TIC.

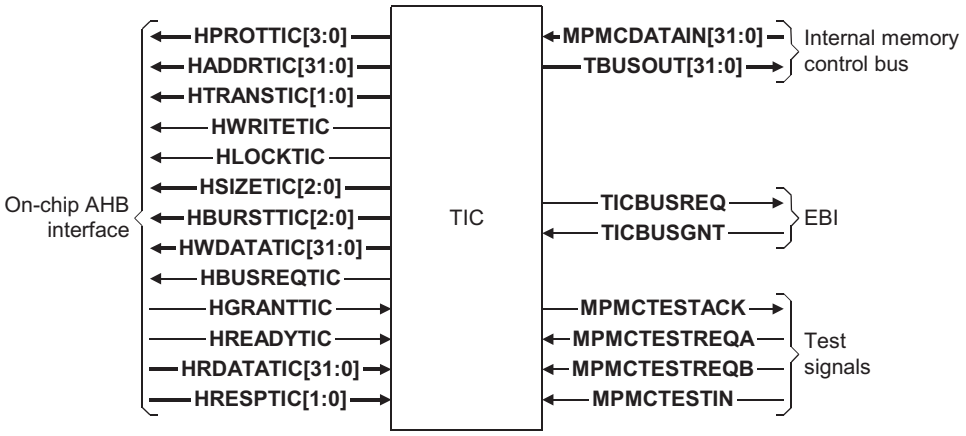


Figure 2-12 TIC block diagram

2.2 Overview of a MPMC, ASIC, or ASSP system

Figure 2-13 shows the MPMC (GX175) in an example system.

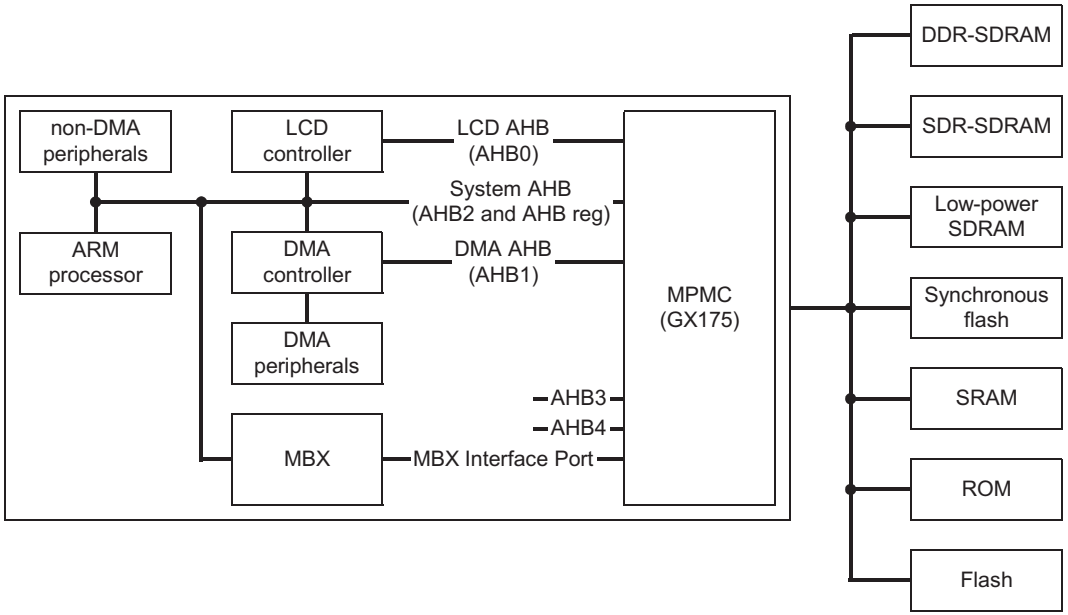


Figure 2-13 MPMC (GX175) in an example system

The example system uses two types of buses as described in:

- *External bus*
- *Internal bus* on page 2-31.

2.2.1 External bus

The off-chip bus that contains data, address, and control signals, connects the ASIC or ASSP to the external memory.

Note

- Connecting a large number of memory devices externally impacts on performance because of signal loading.
- Only one memory device can be accessed at a time.

2.2.2 Internal bus

The on-chip bus enables communication between the on-chip peripherals. The MPMC appears as a standard slave on the on-chip bus and controls the memory on the external bus.

Providing multiple AHB interfaces improves system performance by enabling several access requests to be presented to the memory controller at the same time. This enables the MPMC to pipeline many of the operations (for example, bank activate and precharge), and so reduce the average system access latency and improve utilization of external memory. The use of multiple AHB interfaces also improves system performance by removing heavy DMA traffic from the main AHB bus.

2.3 Low-power operation

In many systems, the contents of the memory system have to be maintained during low-power sleep modes. The MPMC provides two features to enable this:

- dynamic memory refresh over soft reset
- a mechanism to place the dynamic memories into self-refresh mode.

Self-refresh mode can be entered automatically by hardware or manually by software:

- It can be entered manually setting the SREFREQ bit in the MPMCDynamicControl register and polling the SREFACK bit in the MPMCStatus register.
- It can be entered automatically using a *Power Management Unit* (PMU). This is typically present to control the safe transition between the following modes:
  - power-up
  - reset
  - normal
  - sleep.

The PMU can be used to enable self-refresh mode to be entered automatically. To do this, the PMU asserts the **MPMCSREFREQ** signal when self-refresh mode is to be entered. The memory controller then closes any open memory banks and puts the external memory into self-refresh mode. The MPMC then asserts the **MPMCSREFACK** signal to indicate to the PMU that self-refresh mode is entered. The system must ensure that the memory subsystem is idle before asserting **MPMCSREFREQ**. Any transactions to memory that are generated while the memory controller is in self-refresh mode are rejected and an error response is generated (**HRESP** = ERROR). Deasserting **MPMCSREFREQ** returns the memory to normal operation. See the memory data sheet for refresh requirements.

Note

- If **MPMCSREFREQ** is not required this signal must be tied LOW.
- Static memory can be accessed as normal when the SDRAM memory is in self-refresh mode.

2.3.1 Low-power SDRAM deep sleep mode

The MPMC supports JEDEC low-power SDRAM deep-sleep mode. Deep sleep mode can be entered by setting the deep-sleep mode (DP) bit in the MPMCDynamicControl register. The device is then put into a low-power mode where the device is powered down and no longer refreshed. All data in the memory is lost.

2.3.2 Low-power SDRAM partial array refresh

The MPMC supports JEDEC low-power SDRAM partial array refresh. Partial array refresh can be programmed by initializing the SDRAM memory device appropriately. When the memory device is put into self-refresh mode only the memory banks specified are refreshed. The memory banks that are not refreshed lose their data contents.

2.4 Lock and semaphores

Locked accesses on the AHB bus, transactions where **HMASTLOCK** is HIGH, are processed appropriately. When the memory controller performs a locked transfer the memory controller does not rearbitrate to another AHB port until the lock transfer is completed.

2.5 Burst types

All AHB burst types are supported.

Note

INCR transfers are split internally into four-word bursts.

2.6 Busy transfer type

When an AHB master generates busy (AHB **HTRANS** = BUSY) cycles the memory controller waits until busy is inactive before completing the transfer. This increases transfer latency. The memory controller does not rearbitrate to another AHB port if busy goes active.

2.7 Arbitration

The MPMC is normally used with multi-level AHB and no arbitration is required. This section describes the MPMC AHB memory port arbitration strategy when it is used.

Note

- It is recommended that the AHB arbiter, where present, re-arbitrates on a burst boundary because this leads to the most efficient bus utilization.
- AHB protocol does not enable AHB masters to break a defined length burst transfer (INCR4, WRAP4, INCR8, WRAP8, INCR16, and WRAP16). An AHB master can only break an undefined length burst transfer (INCR).
- It is not possible for an AHB burst to cross an SDRAM column, because AHB bursts must not cross a 1KB boundary, and the smallest SDRAM column length supported is 1KB long.

2.7.1 Re-arbitration occurrence

The re-arbitration occurrence is the time when the MPMC re-arbitrates. The following lists AHB transfers that affect re-arbitration:

- If an 8-bit or 16-bit wide AHB burst is submitted and the AHB port buffers are enabled, the MPMC can re-arbitrate to a different AHB port when data is being written to or read from the buffer.

  Note

  The longest AHB transfer is 16 AHB transfers long (INCR16/WRAP16).

- When a locked transfer (as defined by the AHB **HLOCK** signal) has started the AHB memory port is not arbitrated until the locked transfer has completed and the AHB lock signal is deasserted.
- When an AHB burst transfer has started the AHB memory port is not re-arbitrated until the burst has completed.
- The MPMC does not re-arbitrate to another AHB port if **HTRANS** = BUSY.

2.7.2 Re-arbitration priority

The following lists the re-arbitration scheme:

- Auto-refresh is generated.
- The highest priority AHB memory port is a port where the TimeOut register has timed out. If more than one ports has timed out, the port with the lowest port number is selected. For example, port 0 is selected over port 1.
- If none of the AHB memory ports have timed out then the next highest priority port is for the port where the transaction is to an already open SDRAM memory row.

  If the transaction of more than one port is to an already open SDRAM memory row then the port with the lowest port number is selected.
- For the case where none of the AHB memory ports have timed out and none of the accesses are to open SDRAM memory rows. The memory request to the lowest port number is selected. In other words port 0 is selected over port 1.

  Note

  When a buffered 8-bit wide or 16-bit transfer is in progress the memory controller does not re-arbitrate to a different AHB port if the access is to an unopened SDRAM row.

2.7.3 AHB memory port latency

Each AHB memory port has a programmable TimeOut register. When a memory request is made to the port the value of the counter is loaded. Every cycle where the transaction of the port is not serviced the TimeOut register counts down. When the TimeOut counter reaches zero, the priority of the port is increased. This functionality enables each AHB memory port be programmed with a deterministic latency. This also enables the amount of bandwidth that a port consumes to be indirectly defined.

2.8 Worst case transaction latency

The worst case transaction latency for the highest priority AHB memory port is 58 clock cycles. The worst case latency of the lower priority AHB memory ports is TimeOut + 58 cycles, assuming that another higher priority port has not timed out. These values are based on the assumptions given in the following sections:

- *Worst case transaction latency for the highest priority AHB memory port*
- *Worst case transaction latency for the lower priority AHB memory ports* on page 2-40
- *System factors affecting worst case latency* on page 2-42.

2.8.1 Worst case transaction latency for the highest priority AHB memory port

The following assumptions were made for calculating the worst case transaction latency:

- System factors are ignored in these calculations. For information on system factors see *System factors affecting worst case latency* on page 2-42.
- SDRAM memory latency values are:
  - precharge = 3
  - active = 3
  - CAS = 3
  - auto-refresh (t_{RFC}) = 7.
- The SDRAM memory chip selects have a 32-bit wide data bus.
- There are no devices connected to the static memory chip selects.
- AHB port 0 TimeOut register is programmed to be less than or equal to the slowest transfer. This is normally either a INCR16 or WRAP16 read to an unopened SDRAM page.

Note

Using SDRAM memory chip selects with a 16-bit wide data bus, SDRAM memory with larger latency values, or using slow static memory devices in a system affects the worst case latency.

For the MPMC the worst case latency scenario is as follows:

- A lower priority port is performing an INCR16 read request. The read data is to a different row from the one already opened. Therefore a precharge, activate, and read command must be sent to the SDRAM.

- An auto-refresh is generated after the INCR16 read is submitted.
- The highest priority AHB memory port performs an INCR16 read transaction. The read data is to a different row from the one already opened, therefore a precharge, activate, and read command must be sent to the SDRAM.

The read data is to unopened SDRAM memory rows.

Before the first data from the highest priority AHB memory port is returned the following transactions occur:

1. The INCR16 read is performed. The read data is to a different row from the one already opened. This transaction takes 30 cycles.
2. The SDRAM auto-refresh is generated. This transaction takes 13 cycles.
3. The AHB memory port 0 priority read transaction is performed. The read data is to a different row from the one already opened. The first data is returned after 15 cycles.

Worst case transaction latency for highest priority AHB memory port = 30+13+15 = 58 cycles (first data returned).

Worst case latency for highest priority AHB memory port to complete = 30+13+15+15 = 73 cycles (last piece of longest AHB data returned).

2.8.2 Worst case transaction latency for the lower priority AHB memory ports

The following assumptions were made for calculating the worst case transaction latency:

- System factors are ignored in these calculations. For information on system factors see *System factors affecting worst case latency* on page 2-42.
- SDRAM memory latency values are:
  - precharge = 3
  - active = 3
  - CAS = 3
  - auto-refresh (t_{RFC}) = 7.
- The SDRAM memory chip selects have a 32-bit wide data bus.
- There are no devices connected to the static memory chip selects.
- The required AHB port latency is programmed into the appropriate TimeOut register. The value programmed is normally the required time-out latency minus the latency of the slowest burst transfer.

- No other AHB ports have their TimeOut registers programmed.

Note

Using SDRAM memory chip selects with a 16-bit wide data bus, SDRAM memory with larger latency values, or using slow static memory devices in a system affects the worst case latency.

For the MPMC the worst case latency for the lower priority memory port scenario is as follows:

- The AHB memory port that you are interested in submits a read transaction. However this transaction is not performed because there are higher priority transactions. The read data is to a different row from the one already opened. Therefore a precharge, activate, and read command must be sent to the SDRAM.
- An INCR16 read is submitted to a higher priority port. The read data is to unopened SDRAM memory rows.
- The TimeOut register for the AHB memory port that you are interested in times out.
- An auto-refresh is generated after the INCR16 read is submitted.
- The AHB memory port that you are interested in performs a read access. The read data is to a different row from the one already opened. Therefore a precharge, activate, and read command must be sent to the SDRAM.

The following transactions occur before the first data is returned from the port you are interested in:

1. Read transaction submitted.
2. The INCR16 read is performed for higher priority port. The read data is to a different row from the one already opened. This transaction takes 30 cycles.
3. The TimeOut counter for the port you are interested in counts to 0.
4. The SDRAM auto-refresh is generated. This transaction takes 13 cycles.
5. The highest priority read transaction is performed. The read is to an unopened SDRAM memory row. The first data is returned after 15 cycles.

Worst case transaction latency for highest priority AHB memory port = 30+13+15 = 58 cycles.

Worst case latency for highest priority AHB memory port to complete = 30+13+15+15 = 73 cycles.

Note

The latency for the MBX Interface port is similar to that for the lower AHB ports but there is no TimeOut Register available.

2.8.3 System factors affecting worst case latency

This section describes the system factors that can affect the memory controller worst case latency to a memory request.

MPMC AHB Memory port priority

The memory controller AHB memory ports are prioritized so that the most efficient transfers are performed first. High priority ports, AHB ports with a low value, are selected in preference to lower priority ports. The AHB TimeOut register enables the maximum latency for a port to be programmed.

AHB bus

If there are multiple AHB masters on the same AHB bus, then a master can only receive ownership of the bus, and submit transactions, when the AHB arbiter grants the bus to the master. The arbitration strategy in the AHB arbiter then affects the worst case latency for a system with multiple masters on an AHB bus.

AHB masters

When an AHB master generates busy (AHB **HTRANS** = BUSY) cycles the memory controller waits until busy is inactive before completing the transfer. This increases transfer latency. The memory controller does not rearbitrate to another AHB port if busy goes active.

If an AHB master performs locked (AHB **HMASTLOCK** = LOCK) cycles this means that the memory controller cannot rearbitrate until the locked transaction has completed. This can then increase worst case transfer latency, because the memory controller is not able to rearbitrate to a higher priority port until the locked access has completed.

Memory devices

If slow SDRAM memories and/or a 16-bit SDRAM chip select is used, then transactions take longer to complete. This affects transfer latency.

Pad multiplexing

If the memory bus is multiplexed externally, for example, by using an EBI, then the worst case transfer latency is affected because the external bus is shared by multiple devices.

2.9 Sharing memory bandwidth between AHB ports

By programming the AHB port TimeOut values appropriately the bandwidth of the memory controller can be shared between the AHB ports of the memory controller.

Note

The MBX Interface port does not have a TimeOut Register so the AHB ports must be configured so that the MBX Interface port meets the bandwidth requirement.

2.9.1 Typical AHB port TimeOut value given bandwidth requirement

To meet a given bandwidth requirement the TimeOut value must be programmed less than the value provided by the following formula:

$$\text{TimeOut value} = \frac{\text{(AHB frequency) x (number of bytes in average AHB burst)}}{\text{(required AHB bandwidth)}} - \text{(number of transactions in AHB burst)}$$

Note

This formula assumes that the AHB masters on each of the AHB ports always have requests pending.

2.9.2 Typical AHB port TimeOut value given percentage of memory bandwidth requirement

To compute the TimeOut values that must be programmed for the case where each AHB port requires a percentage of the total memory bandwidth, first the expected memory bandwidth of the system must be calculated. From this the bandwidth per port can be calculated. Finally you can use the formula in *Typical AHB port TimeOut value given bandwidth requirement*.

2.9.3 Typical AHB bandwidth requirement example

This section provides an example of how to program the AHB port TimeOut registers given multiple AHB ports and their bandwidth requirements.

System

The memory controller, AHB bus, and SDR-SDRAM memory operate at 100MHz. The SDR-SDRAM has a 32-bit wide data bus.

Bandwidth requirement

The AHB bandwidth requirements are:

- AHB port 0 requires at least 40MB/s
- AHB port 1 requires at least 20MB/s
- AHB port 2 requires at least 2MB/s.

The AHB burst types are:

- AHB port 0 normally performs AHB INCR16, 32-bit wide transfers
- AHB port 1 normally performs AHB INCR8, 32-bit wide transfers
- AHB port 2 normally performs AHB INCR4, 16-bit wide transfers.

Calculations required

The TimeOut values can be calculated as indicated below.

AHB port 0 TimeOut value

Each INCR16 transfer provides 64 bytes of data. Therefore there are 625000 (40MB divided by 64B) INCR16 bursts required per second to satisfy the minimum bandwidth requirement. On average there are 160 clock cycles (100M clock cycles divided by 625000) between AHB port 0 transactions to meet the requirement. Each transaction takes about 16 cycles to complete.

$$\text{TimeOut value} = \frac{100\text{MHz} \times 64}{40\text{MBs}} - 16 = 144 \text{ cycles}$$

Therefore the AHB TimeOut value for AHB port 0 must be programmed to less than 144 clock cycles.

AHB port 1 TimeOut value

Each INCR8 transfer provides 32 bytes of data. Therefore there are 625000 (20MB/32B) INCR8 bursts required per second to satisfy the minimum bandwidth requirement. On average there are 160 clock cycles (100M clock cycles divided by 625000) between AHB port 1 transactions to meet the requirement. Each transaction takes approximately eight cycles to complete.

$$\text{TimeOut value} = \frac{100\text{MHz} \times 32}{20\text{MBs}} - 8 = 152 \text{ cycles}$$

Therefore the AHB TimeOut value for AHB port 1 must be programmed to less than 152 clock cycles.

AHB port 2 TimeOut value

Each INCR4 transfer provides eight bytes of data. Therefore there are 250000 (2MB divided by 8B) INCR4 bursts required per second to satisfy the minimum bandwidth requirement. On average there are 400 clock cycles (100M clock cycles divided by 250000) between AHB port two transactions to meet the requirement. Each transaction takes about four cycles to complete.

$$\text{TimeOut value} = \frac{100\text{MHz} \times 8}{2\text{MBs}} - 4 = 396 \text{ cycles}$$

Therefore the AHB TimeOut value for AHB port 2 must be programmed to less than 396 clock cycles.

2.9.4 Typical AHB percentage bandwidth example

This section provides an example of how to program the AHB port TimeOut registers given multiple AHB ports and their percentage of bandwidth requirements.

System

The memory controller, AHB bus, and SDR-SDRAM memory operate at 100MHz. The SDR-SDRAM has a 32-bit wide data bus.

Bandwidth requirement

The AHB bandwidth requirements are:

- AHB port 0 requires 20% of memory bandwidth minimum
- AHB port 1 requires 10% of memory bandwidth minimum
- AHB port 2 requires 1% of memory bandwidth minimum.

The AHB burst types are:

- AHB port 0 normally performs AHB INCR16, 32-bit wide transfers
- AHB port 1 normally performs AHB INCR8, 32-bit wide transfers
- AHB port 2 normally performs AHB INCR4, 16-bit wide transfers.

Calculations required

You can calculate the TimeOut values as indicated below.

Compute the real system bandwidth

Theoretically the SDRAM memory can provide 400MB/s. However, we assume that the real system bandwidth is 200MB/s because of the system burst types and page hit rates.

Calculate the memory bandwidth per AHB port

For AHB port 0 20% of the 200MB/s memory controller bandwidth is 40MB/s.

For AHB port 1 10% of the 200MB/s memory controller bandwidth is 20MB/s.

For AHB port 2 1% of the 200MB/s memory controller bandwidth is 2MB/s.

Compute the TimeOut value using the following formula

$$\text{TimeOut value} = \frac{\text{(AHB frequency) x (number of bytes in average AHB burst)}}{\text{(required AHB bandwidth)}} - \text{(number of transactions in AHB burst)}$$

For the AHB port 0 the TimeOut value is:

$$\text{TimeOut value} = \frac{100\text{MHz x } 64}{40\text{MBs}} - 16 = 144 \text{ cycles}$$

For the AHB port 1 the TimeOut value is:

$$\text{TimeOut value} = \frac{100\text{MHz x } 32}{20\text{MBs}} - 8 = 152 \text{ cycles}$$

For the AHB port 2 the TimeOut value is:

$$\text{TimeOut value} = \frac{100\text{MHz x } 8}{2\text{MBs}} - 4 = 396 \text{ cycles}$$

2.10 Memory bank select

Eight independently configurable memory chip selects are supported, with a separate AMBA AHB select, **HSELMPMCxCS[7:0],** to select the appropriate chip select:

- **HSELMPMCxCS** selects 0-3 are used to select static memory devices
- **HSELMPMCxCS** selects 4-7 are used to select dynamic memory devices.

The MBX Interface port only requires access to dynamic memory devices using input signals **GSELCS[7:4]**.

Table 2-12 shows the relationship between the AHB select, **HSELMPMCxCS[7:0]**, and the memory chip selects. In the table an x refers to the AHB port number.

Table 2-12 Memory bank selection

AHB select	Chip select	Memory device	Memory chip select
HSELMPMCxCS[0]	0	Static Memory 0	**nMPMCSTCSOUT[0]**
HSELMPMCxCS[1]	1	Static Memory 1	**nMPMCSTCSOUT[1]**
HSELMPMCxCS[2]	2	Static Memory 2	**nMPMCSTCSOUT[2]**
HSELMPMCxCS[3]	3	Static Memory 3	**nMPMCSTCSOUT[3]**
HSELMPMCxCS[4]	4	Dynamic Memory 0	**nMPMCDYCSOUT[0]**
HSELMPMCxCS[5]	5	Dynamic Memory 1	**nMPMCDYCSOUT[1]**
HSELMPMCxCS[6]	6	Dynamic Memory 2	**nMPMCDYCSOUT[2]**
HSELMPMCxCS[7]	7	Dynamic Memory 3	**nMPMCDYCSOUT[3]**

The address range allocated to a chip select is determined by the system AHB decoder.

Note

The largest amount of memory available for a single chip select is 256MB.

2.11 Memory map

The MPMC provides hardware support for booting from external nonvolatile memory. During booting the nonvolatile memory must be located at address `0x00000000` in memory. When the system is booted the SRAM or SDRAM memory can be remapped to address `0x00000000` by modifying the address map in the AHB decoder.

2.11.1 Chip select 1 static memory configuration

The memory width, chip select polarity, and byte lane select polarity of static memory chip select 1 can be configured by using a number of input tie-off signals. This enables you to boot from chip select 1.

The configuration tie-off signals are:

- **MPMCSTCS1MW[1:0]**, memory width select
- **MPMCSTCS1POL**, chip select polarity
- **MPMCSTCS1PB**, byte lane select polarity.

2.11.2 Chip select 5 SDRAM memory configuration

The address mapping and memory device type of SDRAM memory chip select 5 can be configured by using a number of input tie-off signals. This enables you to boot from chip select 5.

The configuration tie-off signals are:

- **MPMCDYCS5ADRMAP[7:0]**, address mapping
- **MPMCDYCS5DEVICE[2:0]**, memory device
- **MPMCDYCS5CASDLY[3:0]**, CAS delay for chip select 5
- **MPMCDYSDRPOL**, polarity of the flip-flop in which the read data is first captured
- **MPMCDYSDRDLY[1:0]**, SDR clocking scheme for data capture.

2.11.3 Boot from flash, SRAM remapped after boot

The system set up is:

- chip select 1 is connected to the boot flash device
- chip select 0 is connected to the SRAM to be remapped to `0x00000000` after boot
- the AHB decoder contains the functionality to remap the address.

The boot sequence is as follows:

1. At power on the reset chip select 1 is located at `0x00000000` in the AHB address map. The following signals are configured so that the nonvolatile memory device can be accessed:
   - **MPMCSTCS1MW[1:0]**
   - **MPMCSTCS1POL** (also **MPMCSTCS0POL**, **MPMCSTCS2POL**, and **MPMCSTCS3POL**)
   - **MPMCSTCS1PB**.
2. When the power-on reset (**nPOR**) and AHB reset (**HRESETn**) go inactive, the processor starts booting from `0x00000000` in memory.
3. The software programs the optimum delay values in the flash memory so that the boot code can run at full speed.
4. The other chip selects are initialized.
5. The AHB decoder address map is modified so that the SRAM is located at address `0x00000000`.
6. The ARM reset and interrupt vectors are copied from flash memory to SRAM that can then be accessed at address `0x00000000`.
7. More boot, initialization, or application code is executed.

2.11.4 Example of a boot from flash, SDRAM remapped after boot

The system set up is:

- chip select 1 is connected to the boot flash device
- chip select 4 is connected to the SDRAM to be remapped to `0x00000000` after boot
- the AHB decoder contains the functionality to remap the address.

The boot sequence is as follows:

1. At power on the reset chip select 1 is located at `0x00000000` in the AHB address map. The following signals are configured so that the nonvolatile memory device can be accessed:
   - **MPMCSTCS1MW[1:0]**
   - **MPMCSTCS1POL** (also **MPMCSTCS0POL**, **MPMCSTCS2POL**, and **MPMCSTCS3POL**)
   - **MPMCSTCS1PB**.
2. When the power-on reset (**nPOR**) and AHB reset (**HRESETn**) go inactive, the processor starts booting from `0x00000000` in memory.
3. The software programs the optimum delay values in the flash memory so that the boot code can run at full speed.
4. The other chip selects are initialized.
5. The AHB decoder address map is modified so that the SDRAM is located at address `0x00000000`.
6. The ARM reset and interrupt vectors are copied from flash memory to SDRAM that can then be accessed at address `0x00000000`.
7. More boot, initialization, or application code is executed.

2.11.5 Memory aliasing

Memory aliasing is permitted.

2.11.6 Unused AHB HADDRx address bits

If some of the **HADDRx** address bits are not required, the unused address bits must be tied LOW.

2.12 Sharing memory interface signals

The memory interface signals, and the memory controller primary input and output signals, can be shared with other peripherals by using an *External Bus Interface* (EBI) block. For more information on the EBI see the *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual*.

Chapter 3
Programmer's Model

This chapter describes the MPMC (GX175) registers and provides details required when programming the microcontroller. It contains the following sections:

3.1 About the programmer's model

The base address of the MPMC is not fixed, but is determined by the AHB decoder, and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed. The registers of the MPMC can only be accessed using the AHB register interface port of the memory controller.

The external memory is accessed using the AHB memory interface ports. Addresses are not fixed, but are determined by the AHB decoder, and can be different for any particular system implementation. Transfers to the external memories of the MPMC are selected by the **HSELMPMC[4:0]CS[7:0]** signals for the AHB ports, and the **GSELCS[7:4]** signals for the MBX interface port.

Note

[4:0] indicates the AHB port number, and [7:0] indicates the chip select to be accessed.

3.2 Register summary

The MPMC (GX175) registers are summarized in Table 3-1.

Table 3-1 MPMC register summary

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCControl	0x000	RW	0x1	0x1	See *Control Register, MPMCControl* on page 3-10
MPMCStatus	0x004	RO	-	0x5	See *Status Register, MPMCStatus* on page 3-11
MPMCConfig	0x008	RW	-	0x-[a]	See *Configuration Register, MPMCConfig* on page 3-11
MPMCDynamicControl	0x020	RW	-	0xE	See *Dynamic Memory Control Register, MPMCDynamicControl* on page 3-12
MPMCDynamicRefresh	0x024	RW	-	0x0	See *Dynamic Memory Refresh Timer Register, MPMCDynamicRefresh* on page 3-15
MPMCDynamicReadConfig	0x028	RW	-	0x----[a]	See *Dynamic Memory Read Configuration Register, MPMCDynamicReadConfig* on page 3-16
MPMCDynamictRP	0x030	RW	-	0xF	See *Dynamic Memory Precharge Command Period Register, MPMCDynamictRP* on page 3-18
MPMCDynamictRAS	0x034	RW	-	0xF	See *Dynamic Memory Active To Precharge Command Period Register, MPMCDynamictRAS* on page 3-18
MPMCDynamictSREX	0x038	RW	-	0x7F	See *Dynamic Memory Self-refresh Exit Time Register, MPMCDynamictSREX* on page 3-19
MPMCDynamictWR	0x044	RW	-	0xF	See *Dynamic Memory Write Recovery Time Register, MPMCDynamictWR* on page 3-20
MPMCDynamictRC	0x048	RW	-	0x1F	See *Dynamic Memory Active To Active Command Period Register, MPMCDynamictRC* on page 3-20

Table 3-1 MPMC register summary (continued)

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCDynamictRFC	0x04C	RW	-	0x1F	See *Dynamic Memory Auto-refresh Period Register, MPMCDynamictRFC* on page 3-21
MPMCDynamictXSR	0x050	RW	-	0xFF	See *Dynamic Memory Exit Self-refresh Register, MPMCDynamictXSR register* on page 3-22
MPMCDynamictRRD	0x054	RW	-	0xF	See *Dynamic Memory Active Bank A To Active Bank B Time Register, MPMCDynamictRRD* on page 3-22
MPMCDynamictMRD	0x058	RW	-	0xF	See *Dynamic Memory Load Mode Register, MPMCDynamictMRD register* on page 3-23
MPMCDynamictCDLR	0x05C	RW	-	0xF	See *Dynamic Memory Last Data In To Read Command Time Register, MPMCDynamictCDLR* on page 3-24
MPMCStaticExtendedWait	0x080	RW	-	0x0	See *Static Memory Extended Wait Register, MPMCStaticExtendedWait* on page 3-24
MPMCDynamicConfig0	0x100	RW	-	0x0	See *Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3* on page 3-25
MPMCDynamicRasCas0	0x104	RW	-	0x783	See *Dynamic Memory RAS and CAS Delay Registers 0-3, MPMCDynamicRasCas0-3* on page 3-30
MPMCDynamicConfig1	0x120	RW	-	0x0--[a]	See *Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3* on page 3-25
MPMCDynamicRasCas1	0x124	RW	-	0x--3[a]	See *Dynamic Memory RAS and CAS Delay Registers 0-3, MPMCDynamicRasCas0-3* on page 3-30
MPMCDynamicConfig2	0x140	RW	-	0x0	See *Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3* on page 3-25

Table 3-1 MPMC register summary (continued)

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCDynamicRasCas2	0x144	RW	-	0x783	See *Dynamic Memory RAS and CAS Delay Registers 0-3, MPMCDynamicRasCas0-3* on page 3-30
MPMCDynamicConfig3	0x160	RW	-	0x0	See *Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3* on page 3-25
MPMCDynamicRasCas3	0x164	RW	-	0x783	See *Dynamic Memory RAS and CAS Delay Registers 0-3, MPMCDynamicRasCas0-3* on page 3-30
MPMCStaticConfig0	0x200	RW	-	0x-0[a]	See *Static Memory Configuration Registers 0-3, MPMCStaticConfig0-3* on page 3-31
MPMCStaticWaitWen0	0x204	RW	-	0x0	See *Static Memory Write Enable Delay Registers 0-3, MPMCStaticWaitWen0-3* on page 3-33
MPMCStaticWaitOen0	0x208	RW	-	0x0	See *Static Memory Output Enable Delay Registers 0-3, MPMCStaticWaitOen0-3* on page 3-34
MPMCStaticWaitRd0	0x20C	RW	-	0x1F	See *Static Memory Read Delay Registers 0-3, MPMCStaticWaitRd0-3* on page 3-34
MPMCStaticWaitPage0	0x210	RW	-	0x1F	See *Static Memory Page Mode Read Delay Registers 0-3, MPMCStaticWaitPage0-3* on page 3-35
MPMCStaticWaitWr0	0x214	RW	-	0x1F	See *Static Memory Write Delay Registers 0-3, MPMCStaticWaitWr0-3* on page 3-36
MPMCStaticWaitTurn0	0x218	RW	-	0xF	See *Static Memory Turn Round Delay Registers 0-3, MPMCStaticWaitTurn0-3* on page 3-36
MPMCStaticConfig1	0x220	RW	-	0x--[a]	See *Static Memory Configuration Registers 0-3, MPMCStaticConfig0-3* on page 3-31
MPMCStaticWaitWen1	0x224	RW	-	0x0	See *Static Memory Write Enable Delay Registers 0-3, MPMCStaticWaitWen0-3* on page 3-33

Table 3-1 MPMC register summary (continued)

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCStaticWaitOen1	0x228	RW	-	0x0	See *Static Memory Output Enable Delay Registers 0-3, MPMCStaticWaitOen0-3* on page 3-34
MPMCStaticWaitRd1	0x22C	RW	-	0x1F	See *Static Memory Read Delay Registers 0-3, MPMCStaticWaitRd0-3* on page 3-34
MPMCStaticWaitPage1	0x230	RW	-	0x1F	See *Static Memory Page Mode Read Delay Registers 0-3, MPMCStaticWaitPage0-3* on page 3-35
MPMCStaticWaitWr1	0x234	RW	-	0x1F	See *Static Memory Write Delay Registers 0-3, MPMCStaticWaitWr0-3* on page 3-36
MPMCStaticWaitTurn1	0x238	RW	-	0xF	See *Static Memory Turn Round Delay Registers 0-3, MPMCStaticWaitTurn0-3* on page 3-36
MPMCStaticConfig2	0x240	RW	-	0x-0[a]	See *Static Memory Configuration Registers 0-3, MPMCStaticConfig0-3* on page 3-31
MPMCStaticWaitWen2	0x244	RW	-	0x0	See *Static Memory Write Enable Delay Registers 0-3, MPMCStaticWaitWen0-3* on page 3-33
MPMCStaticWaitOen2	0x248	RW	-	0x0	See *Static Memory Output Enable Delay Registers 0-3, MPMCStaticWaitOen0-3* on page 3-34
MPMCStaticWaitRd2	0x24C	RW	-	0x1F	See *Static Memory Read Delay Registers 0-3, MPMCStaticWaitRd0-3* on page 3-34
MPMCStaticWaitPage2	0x250	RW	-	0x1F	See *Static Memory Page Mode Read Delay Registers 0-3, MPMCStaticWaitPage0-3* on page 3-35
MPMCStaticWaitWr2	0x254	RW	-	0x1F	See *Static Memory Write Delay Registers 0-3, MPMCStaticWaitWr0-3* on page 3-36
MPMCStaticWaitTurn2	0x258	RW	-	0xF	See *Static Memory Turn Round Delay Registers 0-3, MPMCStaticWaitTurn0-3* on page 3-36
MPMCStaticConfig3	0x260	RW	-	0x-0[a]	See *Static Memory Configuration Registers 0-3, MPMCStaticConfig0-3* on page 3-31

Table 3-1 MPMC register summary (continued)

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCStaticWaitWen3	0x264	RW	-	0x0	See *Static Memory Write Enable Delay Registers 0-3, MPMCStaticWaitWen0-3* on page 3-33
MPMCStaticWaitOen3	0x268	RW	-	0x0	See *Static Memory Output Enable Delay Registers 0-3, MPMCStaticWaitOen0-3* on page 3-34
MPMCStaticWaitRd3	0x26C	RW	-	0x1F	See *Static Memory Read Delay Registers 0-3, MPMCStaticWaitRd0-3* on page 3-34
MPMCStaticWaitPage3	0x270	RW	-	0x1F	See *Static Memory Page Mode Read Delay Registers 0-3, MPMCStaticWaitPage0-3* on page 3-35
MPMCStaticWaitWr3	0x274	RW	-	0x1F	See *Static Memory Write Delay Registers 0-3, MPMCStaticWaitWr0-3* on page 3-36
MPMCStaticWaitTurn3	0x278	RW	-	0xF	See *Static Memory Turn Round Delay Registers 0-3, MPMCStaticWaitTurn0-3* on page 3-36
MPMCAHBControl0	0x400	RW	-	0x0	See *AHB Control Registers 0-4, MPMCAHBControl0-4* on page 3-37
MPMCAHBStatus0	0x404	RW	-	0x0	See *AHB Status Registers 0-4, MPMCAHBStatus0-4* on page 3-39
MPMCAHBTimeOut0	0x408	RW	-	0x0	See *AHB TimeOut Registers 0-4, MPMCAHBTimeOut0-4* on page 3-39
MPMCAHBControl1	0x420	RW	-	0x0	See *AHB Control Registers 0-4, MPMCAHBControl0-4* on page 3-37
MPMCAHBStatus1	0x424	RW	-	0x0	See *AHB Status Registers 0-4, MPMCAHBStatus0-4* on page 3-39
MPMCAHBTimeOut1	0x428	RW	-	0x0	See *AHB TimeOut Registers 0-4, MPMCAHBTimeOut0-4* on page 3-39
MPMCAHBControl2	0x440	RW	-	0x0	See *AHB Control Registers 0-4, MPMCAHBControl0-4* on page 3-37
MPMCAHBStatus2	0x444	RW	-	0x0	See *AHB Status Registers 0-4, MPMCAHBStatus0-4* on page 3-39

Table 3-1 MPMC register summary (continued)

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCAHBTimeOut2	0x448	RW	-	0x0	See *AHB TimeOut Registers 0-4, MPMCAHBTimeOut0-4* on page 3-39
MPMCAHBControl3	0x460	RW	-	0x0	See *AHB Control Registers 0-4, MPMCAHBControl0-4* on page 3-37
MPMCAHBStatus3	0x464	RW	-	0x0	See *AHB Status Registers 0-4, MPMCAHBStatus0-4* on page 3-39
MPMCAHBTimeOut3	0x468	RW	-	0x0	See *AHB TimeOut Registers 0-4, MPMCAHBTimeOut0-4* on page 3-39
MPMCAHBControl4	0x480	RW	-	0x0	See *AHB Control Registers 0-4, MPMCAHBControl0-4* on page 3-37
MPMCAHBStatus4	0x484	RW	-	0x0	See *AHB Status Registers 0-4, MPMCAHBStatus0-4* on page 3-39
MPMCAHBTimeOut4	0x488	RW	-	0x0	See *AHB TimeOut Registers 0-4, MPMCAHBTimeOut0-4* on page 3-39
MPMCITCR	0xF00	RW	0x-[a]	0x-[a]	See *Test Control Register, MPMCITCR* on page 4-4
MPMCITIP0	0xF20	RW	0x----[a]	0x----[a]	See *Test Input 0 Register, MPMCITIP0* on page 4-5
MPMCITIP1	0xF24	RW	0x--[a]	0x--[a]	See *Test Input 1 Register, MPMCITIP1* on page 4-9
MPMCITOP	0xF40	RW	-	0x1[b]	See *Test Output Register, MPMCITOP* on page 4-10
MPMCPeriphID4	0xFD0	RO	0x5	0x5	See *Additional Peripheral Identification Register 4, MPMCPeriphID4* on page 3-40
MPMCPeriphID5	0xFD4	RO	0x0	0x0	See *Additional Peripheral Identification Registers 5-7, MPMCPeriphID5-7* on page 3-41
MPMCPeriphID6	0xFD8	RO	0x0	0x0	See *Additional Peripheral Identification Registers 5-7, MPMCPeriphID5-7* on page 3-41

Table 3-1 MPMC register summary (continued)

Register	Offset	Type	Reset HRESETn	Reset nPOR	Description
MPMCPeriphID7	0xFDC	RO	0x0	0x0	See *Additional Peripheral Identification Registers 5-7, MPMCPeriphID5-7* on page 3-41
MPMCPeriphID0	0xFE0	RO	0x75	0x75	See *Peripheral Identification Register 0, MPMCPeriphID0* on page 3-42
MPMCPeriphID1	0xFE4	RO	0x11	0x11	See *Peripheral Identification Register 1, MPMCPeriphID1 register* on page 3-43
MPMCPeriphID2	0xFE8	RO	0x-4[c]	0x-4[a]	See *Peripheral Identification Register 2, MPMCPeriphID2 register* on page 3-43
MPMCPeriphID3	0xFEC	RO	0x47	0x47	See *Peripheral Identification Register 3, MPMCPeriphID3 register* on page 3-44
MPMCPCellID0	0xFFO	RO	0xD	0xD	See *PrimeCell Identification Registers 0-3, MPMCPCellID0-3* on page 3-45
MPMCPCellID1	0xFF4	RO	0xF0	0xF0	See *PrimeCell Identification Registers 0-3, MPMCPCellID0-3* on page 3-45
MPMCPCellID2	0xFF8	RO	0x5	0x5	See *PrimeCell Identification Registers 0-3, MPMCPCellID0-3* on page 3-45
MPMCPCellID3	0xFFC	RO	0xB1	0xB1	See *PrimeCell Identification Registers 0-3, MPMCPCellID0-3* on page 3-45

a. Tie-off dependent.
b. Reflects **MPMCSREFACK** that is set on power-on reset.
c. Revision dependent.

3.3 Register descriptions

This section describes the MPMC registers.

3.3.1 Control Register, MPMCControl

The MPMCControl Register is a two-bit, read/write register that controls the memory controller operation. The register fields can only be altered in idle state. This register can be accessed with zero wait states. Table 3-2 lists the bit assignments for the MPMCControl Register.

Table 3-2 MPMCControl Register bit assignments

Bits	Name	Description
[31:3]	-	Reserved, read undefined, do not modify.
[2]	L	Low-power mode, indicates normal, or low-power mode: 0 = normal mode (reset value on **nPOR**, and **HRESETn**) 1 = low-power mode. Entering low-power mode reduces memory controller power consumption. Dynamic memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit (L), or by AHB, or power-on reset. You must only modify this bit when the MPMC is in idle state.[ab]
[1]	-	Reserved, read undefined, do not modify.
[0]	E	MPMC enable, indicates if the MPMC is enabled or disabled: 0 = disabled 1 = enabled (reset value on **nPOR** and **HRESETn**). Disabling the MPMC reduces power consumption. When the memory controller is disabled the memory is not refreshed. The memory controller is enabled by setting the enable bit, or by AHB, or power-on reset. You must only modify this bit when the MPMC is in idle state.[ab]

a. The external memory cannot be accessed in low-power states. If a memory access is performed an error response is generated.
b. The memory controller AHB register programming port can be accessed normally. The MPMC registers can be programmed in low-power and/or disabled state.

3.3.2 Status Register, MPMCStatus

The two-bit read-only MPMCStatus Register provides MPMC status information. This register can be accessed with zero wait states. Table 3-3 lists the bit assignments for the MPMCStatus Register.

Table 3-3 MPMCStatus Register bit assignments

Bits	Name	Description
[31:3]	-	Reserved, read undefined.
[2]	SA	Self-refresh acknowledge, **MPMCSREFACK**. This read-only bit indicates the operating mode of the dynamic memory controller: 0 = normal mode 1 = self-refresh mode (reset value on **nPOR**).
[1]	-	Reserved, read undefined.
[0]	B	Busy, this read-only bit is used to ensure that the memory controller enters the low-power or disabled mode cleanly by determining if the memory controller is busy or not: 0 = MPMC is idle 1 = MPMC is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode (reset value on **nPOR** and **HRESETn**).

3.3.3 Configuration Register, MPMCConfig

The one-bit, read/write, MPMCConfig Register configures the operation of the memory controller. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting

until the MPMC is idle and then entering low-power, or disabled mode. MPMCConfig is accessed with one wait state. Table 3-4 lists the bit assignments for the MPMCConfig Register.

Table 3-4 MPMCConfig Register bit assignments

Bits	Name	Description
[31:1]	-	Reserved, read undefined, do not modify.
[0]	N	Endian mode: 0 = little-endian mode, reset value on **nPOR** 1 = big-endian mode. The MPMC endian mode on **nPOR** is determined by **MPMCBIGENDIAN** but the endian mode selected by the **MPMCBIGENDIAN** hardware input is not reflected by this register bit. This bit only reflects the last value that was written to it by software. You must flush all data in the MPMC before switching between little-endian and big-endian modes. This register bit is unaffected by **HRESETn**.

3.3.4 Dynamic Memory Control Register, MPMCDynamicControl

The fourteen-bit, read/write, MPMCDynamicControl Register is used to control dynamic memory operation. The control bits can be altered during normal operation. This register can be accessed with zero wait states. Table 3-5 lists the bit assignments for the MPMCDynamicControl Register.

Table 3-5 MPMCDynamicControl Register bit assignments

Bits	Name	Description
[31:16]	-	Reserved, read undefined, do not modify.
[15]	RPVHH	SyncFlash reset/power down voltage signal, **MPMCRPVHHOUT**: 0 = normal voltage (reset value on **nPOR**) 1 = set **MPMCRPVHHOUT** high voltage. This output can be used externally in conjunction with the **nMPMCRPOUT** signal to indicate a high output voltage, see Table 3-7 on page 3-15.
[14]	nRP	SyncFlash reset/power down signal, **nMPMCRPOUT**: 0 = **nMPMCRPOUT** signal LOW (reset value on **nPOR**) 1 = set **nMPMCRPOUT** signal HIGH.
[13]	DP	Low-power SDRAM deep-sleep mode: 0 = normal operation (reset value on **nPOR**) 1 = enter deep power down mode.
[12]	-	Reserved, read undefined, do not modify.

Table 3-5 MPMCDynamicControl Register bit assignments (continued)

Bits	Name	Description
[11]	DE	Enable DLL hand shaking, **MPMCDLLCALIREQ**, during auto-refresh cycles: 0 = DLL hand shaking disabled (reset value on **nPOR**) 1 = DLL hand shaking enabled.
[10]	DC	DLL calibrate, software control of the DLL hand shaking, **MPMCDLLCALIREQ**, for initial DLL calibration: 0 = DLL hand shaking disabled (reset value on **nPOR**) 1 = DLL hand shaking enabled.
[9]	DS	DLL status, indicates the value of the **MPMCDLLCALIACK** signal for software control of DLL hand shaking: 0 = DLL calibrating 1 = DLL calibration completed.
[8:7]	I	SDRAM initialization: b00 = issue SDRAM NORMAL operation command (reset value on **nPOR**) b01 = issue SDRAM MODE command b10 = issue SDRAM PALL (precharge all) command b11 = issue SDRAM NOP (no operation) command).
[6]	-	Reserved, read undefined, do not modify.
[5]	MCC	Memory clock control: 0 = **MPMCCLKOUT** enabled (reset value on **nPOR**) 1 = **MPMCCLKOUT** disabled.[a]
[4]	IMCC	Inverted memory clock control: 0 = **nMPMCCLKOUT** enabled (reset value on **nPOR**) 1 = **nMPMCCLKOUT** disabled.[b]
[3]	SRMCC	Self-refresh clock control: 0 = **MPMCCLKOUT** and **nMPMCCLKOUT** stop during self-refresh mode. 1 = **MPMCCLKOUT** and **nMPMCCLKOUT** run continuously (reset value on **nPOR**).[c]

Table 3-5 MPMCDynamicControl Register bit assignments (continued)

Bits	Name	Description
[2]	SR	Self-refresh request, **MPMCSREFREQ**: 0 = normal mode 1 = enter self-refresh mode (reset value on **nPOR**). By writing 1 to this bit self-refresh mode can be entered under software control. Writing 0 to this bit returns the MPMC to normal mode. The self-refresh acknowledge bit in the MPMCStatus register must be polled to determine the current operating mode of the MPMC.
[1]	CS	Dynamic memory clock control: 0 = **MPMCCLKOUT** stops when all SDRAMS are idle 1 = **MPMCCLKOUT** runs continuously (reset value on **nPOR**).[d]
[0]	CE	Dynamic memory clock enable: 0 = clock enable of idle devices are deasserted to save power (reset value on **nPOR**) 1 = all clock enables are driven HIGH continuously.[e]

a. Disabling **MPMCCLKOUT** can be performed if there are no SDRAM memory transactions. When enabled this field can be used in conjunction with the dynamic memory clock control (CS) field.
b. Disabling **nMPMCCLKOUT** can be performed if there are no DDR-SDRAM memory transactions that require a differential clock. When enabled this field can be used in conjunction with the dynamic memory clock control (CS) field.
c. This functionality is supported by SDR-SDRAM and DDR-SDRAM.
d. This functionality is only supported by SDR-SDRAM.
e. Clock enable must be HIGH during SDRAM initialization.

Table 3-6 lists the clock status options, based on the settings of the following bits:

CKE Clock enable.

CS Clock stop (bit 1 in Table 3-5 on page 3-12).

SRMCC Self-refresh clock stop (bit 3 in Table 3-5 on page 3-12).

Table 3-6 Clock status options

CKE	CS	SRMCC	Clock status
1	x	x	Run
0	1	1	Run
0	0	1	Stop
0	0	0	Stop
0	1	0	Stop

Table 3-7 lists the output voltage settings for different combinations of bits [15:14].

To boot from the SyncFlash device, it must be taken out of reset before the MPMC is taken out of reset. This requires some external logic to drive both the **nRP** pin of the SyncFlash and the **MPMCBOOTDLY** signal of the MPMC. In this case, the **nMPMCRPOUT** signal must be routed through the external logic to the **nRP** pin of the SyncFlash device. The delay required between the **nRP** signal going HIGH and the **MPMCBOOTDLY** signal going HIGH is specific to the SyncFlash device. See the applicable data sheet for more information.

If booting from SyncFlash is not required, the **nMPMCRPOUT** signal can directly drive the **nRP** pin of the SyncFlash device. The software should give enough time between taking this pin HIGH, and performing accesses.

A block external to the MPMC can use the values of the **nMPMCRPOUT** and **MPMCRPVHHOUT** signals to generate the required voltage settings for the Micron SyncFlash reset signal. Some systems do not have an 8V output, or do not require the functionality to raise **nRP** to 8V. In these cases **MPMCRPVHHOUT** can be ignored.

Table 3-7 Output voltage settings

nMPMCRPOUT	MPMCRPVHHOUT	Output
0	0	0V
0	1	0V
1	0	3V
1	1	8V

3.3.5 Dynamic Memory Refresh Timer Register, MPMCDynamicRefresh

The 11-bit, read/write, MPMCDynamicRefresh Register configures dynamic memory operation. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. However, these control bits can, if necessary, be altered during normal operation. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-8 lists the bit assignments for the MPMCDynamicRefresh Register.

Table 3-8 MPMCDynamicRefresh Register bit assignments

Bits	Name	Description
[31:11]	-	Reserved, read undefined, do not modify.
[10:0]	REFRESH	Refresh timer: 0x0 = refresh disabled (reset value on **nPOR**) 0x1 1(x16) = 16 clock cycles between SDRAM refresh cycles 0x8 8(x16) = 128 clock cycles between SDRAM refresh cycles 0x1-0x7FF n(x16) = 16n clock cycles between SDRAM refresh cycles.

For example, for the refresh period of 16µs, and an **HCLK** frequency of 50MHz, the following value must be programmed into this register:

$$\frac{16 \times 10^{-6} \times 50 \times 10^{6}}{16} = 50 \text{ or } 0x32$$

Note

The refresh cycles are evenly distributed. However, there might be slight variations when the auto-refresh command is issued depending on the status of the memory controller.

3.3.6 Dynamic Memory Read Configuration Register, MPMCDynamicReadConfig

The six-bit, read/write, MPMCDynamicReadConfig Register enables you to configure the dynamic memory read strategy. This register must only be modified during system initialization. This register can be accessed with one wait state. Table 3-9 lists the bit assignments for the MPMCDynamicReadConfig Register.

Table 3-9 MPMCDynamicReadConfig Register bit assignments

Bits	Name	Description
[31:13]	-	Reserved, read undefined, do not modify.
[12]	DRP	DDR-SDRAM read data capture polarity: 0 = data capture on the negative edge of **HCLK** 1 = data capture on the positive edge of **HCLK**.[a]
[11:10]	-	Reserved, read undefined, do not modify.

Table 3-9 MPMCDynamicReadConfig Register bit assignments (continued)

Bits	Name	Description
[9:8]	DRD	DDR-SDRAM read data strategy: b00 = clock out delayed strategy, using **MPMCCLKOUT** (command not delayed, clock out delayed) b01 = command delayed strategy, using **MPMCCLKDELAY** (command delayed, clock out not delayed) b10 = command delayed strategy plus one clock cycle, using **MPMCCLKDELAY** (command delayed, clock out not delayed) b11 = command delayed strategy plus two clock cycles, using **MPMCCLKDELAY** (command delayed, clock out not delayed).[b]
[7:5]	-	Reserved, read undefined, do not modify.
[4]	SRP	SDR-SDRAM read data capture polarity: 0 = data capture on the negative edge of **HCLK** 1 = data capture on the positive edge of **HCLK**.[c]
[3:2]	-	Reserved, read undefined, do not modify.
[1:0]	SRD	SDR-SDRAM read data strategy: b00 = clock out delayed strategy, using **MPMCCLKOUT** (command not delayed, clock out delayed) b01 = command delayed strategy, using **MPMCCLKDELAY** (command delayed, clock out not delayed) b10 = command delayed strategy plus one clock cycle, using **MPMCCLKDELAY** (command delayed, clock out not delayed) b11 = command delayed strategy plus two clock cycles, using **MPMCCLKDELAY** (command delayed, clock out not delayed).[d]

a. The value of the DRP field on **nPOR** is determined by the **MPMCDYDDRPOL** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.
b. The value of the DRD field on **nPOR** is determined by the **MPMCDYDDRDLY[1:0]** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.
c. The value of the SRP field on **nPOR** is determined by the **MPMCDYSDRPOL** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.
d. The value of the SRD field on **nPOR** is determined by the **MPMCDYSDRDLY[1:0]** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.

3.3.7 Dynamic Memory Precharge Command Period Register, MPMCDynamictRP

The four-bit, read/write, MPMCDynamictRP Register enables you to program the precharge command period, t_{RP}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{RP}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-10 lists the bit assignments for the MPMCDynamictRP Register.

Table 3-10 MPMCDynamictRP Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	t_{RP}	Precharge command period: 0x0-0xE = n+1 clock cycles 0xF = 16 clock cycles (reset value on **nPOR**).

3.3.8 Dynamic Memory Active To Precharge Command Period Register, MPMCDynamictRAS

The four-bit, read/write, MPMCDynamictRAS Register enables you to program the active to precharge command period, t_{RAS}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RAS}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-11 lists the bit assignments for the MPMCDynamictRAS Register.

Table 3-11 MPMCDynamictRAS Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	t_{RAS}	Active to precharge command period: 0x0-0xE = n+1 clock cycles 0xF = 16 clock cycles (reset value on **nPOR**). The chosen value of t_{RAS} clock cycles must be greater than the RAS field in the *MPMCDynamicRasCas0-3 Register bit assignments* on page 3-30. Using a value of t_{RAS} clock cycles less than or equal to RAS will lead to memory access failures.

3.3.9 Dynamic Memory Self-refresh Exit Time Register, MPMCDynamictSREX

The seven-bit, read/write, MPMCDynamictSREX Register enables you to program the self-refresh exit time, t_{SREX}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{SREX}, for devices without this parameter you must use the same value as t_{XSR}. For some DDR-SDRAM data sheets this parameter is known as t_{XSNR}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-12 lists the bit assignments for the MPMCDynamictSREX Register.

Table 3-12 MPMCDynamictSREX Register bit assignments

Bits	Name	Description
[31:7]	-	Reserved, read undefined, do not modify.
[6:0]	t_{SREX}	Self-refresh exit time: 0x0-0x7E = n+1 clock cycles 0x7F = 128 clock cycles (reset value on **nPOR**).

3.3.10 Dynamic Memory Write Recovery Time Register, MPMCDynamictWR

The four-bit, read/write, MPMCDynamictWR Register enables you to program the write recovery time, t_{WR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{WR}, t_{DPL}, t_{RWL}, or t_{RDL}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-13 lists the bit assignments for the MPMCDynamictWR Register.

Table 3-13 MPMCDynamictWR Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	t_{WR}	Write recovery time: 0x0-0xE = n+1 clock cycles 0xF = 16 clock cycles (reset value on **nPOR**).

3.3.11 Dynamic Memory Active To Active Command Period Register, MPMCDynamictRC

The five-bit, read/write, MPMCDynamictRC Register enables you to program the active to active command period, t_{RC}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RC}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-14 lists the bit assignments for the MPMCDynamictRC Register.

Table 3-14 MPMCDynamictRC Register bit assignments

Bits	Name	Description
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	t_{RC}	Active to active command period: 0x0-0x1E = n+1 clock cycles 0x1F = 32 clock cycles (reset value on **nPOR**).

3.3.12 Dynamic Memory Auto-refresh Period Register, MPMCDynamictRFC

The five-bit, read/write, MPMCDynamictRFC Register enables you to program the auto-refresh period, and auto-refresh to active command period, t_{RFC}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RFC}, or sometimes as t_{RC}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-15 lists the bit assignments for the MPMCDynamictRFC Register.

Table 3-15 MPMCDynamictRFC Register bit assignments

Bits	Name	Description
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	t_{RFC}	Auto-refresh period and auto-refresh to active command period: 0x0-0x1E = n+1 clock cycles 0x1F = 32 clock cycles (reset value on **nPOR**).

3.3.13 Dynamic Memory Exit Self-refresh Register, MPMCDynamictXSR register

The eight-bit, read/write, MPMCDynamictXSR Register enables you to program the exit self-refresh to active command time, t_{XSR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{XSR}, but is sometimes called t_{XSNR} in some DDR-SDRAM data sheets. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-16 lists the bit assignments for the MPMCDynamictXSR Register.

Table 3-16 MPMCDynamictXSR Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined, do not modify.
[7:0]	t_{XSR}	Exit self-refresh to active command time: `0x0-0xFE` = n+1 clock cycles `0xFF` = 256 clock cycles (reset value on **nPOR**).

3.3.14 Dynamic Memory Active Bank A To Active Bank B Time Register, MPMCDynamictRRD

The four-bit, read/write, MPMCDynamictRRD Register enables you to program the active bank A to active bank B latency, t_{RRD}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RRD}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-17 lists the bit assignments for the MPMCDynamictRRD Register.

Table 3-17 MPMCDynamictRRD Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	t_{RRD}	Active bank A to active bank B latency: 0x0-0xE = n+1 clock cycles 0xF = 16 clock cycles (reset value on **nPOR**)

3.3.15 Dynamic Memory Load Mode Register, MPMCDynamictMRD register

The four-bit, read/write, MPMCDynamictMRD Register enables you to program the load mode register to active command time, t_{MRD}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{MRD}, or t_{RSA}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-18 lists the bit assignments for the MPMCDynamictMRD Register.

Table 3-18 MPMCDynamictMRD Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	t_{MRD}	Load mode register to active command time: 0x0-0xE = n+1 clock cycles 0xF = 16 clock cycles (reset value on **nPOR**).

3.3.16 Dynamic Memory Last Data In To Read Command Time Register, MPMCDynamictCDLR

The four-bit, read/write, MPMCDynamictCDLR register enables you to program the last data in to read command time, t_{CDLR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{CDLR}. This register can be accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 3-19 lists the bit assignments for the MPMCDynamictCDLR Register.

Table 3-19 MPMCDynamictCDLR Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	t_{CDLR}	Last data in to read command time: `0x0-0xE` = n+1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**).

3.3.17 Static Memory Extended Wait Register, MPMCStaticExtendedWait

The 10-bit, read/write, MPMCStaticExtendedWait Register is used to time long static memory read and write transfers (that are longer than can be supported by the MPMCStaticWaitRd0-3 or, MPMCStaticWaitWr0-3 Registers) when the EW bit of the MPMCStaticConfig Registers is enabled. There is only a single MPMCStaticExtendedWait Register. This is used by the relevant static memory chip select if the appropriate *Extended Wait* (EW) bit in the MPMCStaticConfig Register is set. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. However, if necessary, these control bits can be altered during normal operation. This register can be accessed with one wait state.

Table 3-20 lists the bit assignments for the MPMCStaticExtendedWait Register.

Table 3-20 MPMCStaticExtendedWait Register bit assignments

Bits	Name	Description
[31:10]	-	Reserved, read undefined, do not modify.
[9:0]	EXTENDEDWAIT	External wait time-out: 0x0 = 16 clock cycles (reset value on **nPOR**) 0x1-0x3FF = (n+1) x16 clock cycles.

For example, for a static memory read or write transfer time of 16µs, and an **HCLK** frequency of 50MHz, the following value must be programmed into this register:

$$\frac{16 \times 10^{-6} \times 50 \times 10^{6}}{16} - 1 = 49 \text{ or } 0x31$$

3.3.18 Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3

The 12-bit, read/write, MPMCDynamicConfig0-3 Registers enable you to program the configuration information for the relevant dynamic memory chip select. These registers are normally only modified during system initialization. These registers can be accessed with one wait state.

Table 3-21 lists the bit assignments for the MPMCDynamicConfig0-3 Registers.

Table 3-21 MPMCDynamicConfig0-3 Register bit assignments

Bits	Name	Description
[31:21]	-	Reserved, read undefined, do not modify.
[20]	P	Write protect: 0 = writes not protected (reset value on **nPOR**) 1 = write-protected.
[19:15]	-	Reserved, read undefined, do not modify.

Table 3-21 MPMCDynamicConfig0-3 Register bit assignments (continued)

Bits	Name	Description
[14:7]	AM	Address mapping, see Table 3-22 on page 3-27. b00000000 = reset value on **nPOR**.[a] The value of the chip select 5 address mapping field on power-on reset (**nPOR**) is determined by the **MPMCDYCS5ADRMAP[7:0]** signal. This value can be overridden by software. This field is unaffected by AHB reset (**HRESETn**).[b]
[6:3]	-	Reserved, read undefined, do not modify.
[2:0]	MD	Memory device: b000 = SDR-SDRAM (reset value on **nPOR**) b001 = SDR Micron SyncFlash b010 = low-power SDR-SDRAM b011 = SDR Micron V-SyncFlash b100 = DDR-SDRAM b101 = DDR Micron SyncFlash b110 = low-power DDR-SDRAM b111 = DDR Micron V-SyncFlash. The value of the chip select 5 memory device field on power-on reset (**nPOR**) is determined by the **MPMCDYCS5DEVICE[2:0]** signal. This value can be overridden by software. This field is unaffected by AHB reset (**HRESETn**).[c]

a. The SDRAM column and row width and number of banks are computed automatically from the address mapping.
b. The value of the **MPMCDYCS5ADRMAP[7:0]** signal is reflected in this field. When programmed this register reflects the last value that is written into it.
c. The value of the **MPMCDYCS5DEVICE[2:0]** signal is reflected in this field. When programmed this register reflects the last value that is written into it.

Note

DDR-SDRAM device chip selects must only have a 16-bit wide data bus.

Address mappings that are not shown in Table 3-22 are reserved.

Table 3-22 Address mapping

[14]	[13:12]	[11:9]	[8:7]	Description
16-bit external bus high-performance address mapping (Row, Bank, Column)				
0	b00	b000	b00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	b00	b000	b01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	b00	b001	b00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	b00	b001	b01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	b00	b010	b00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	b00	b010	b01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	b00	b011	b00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	b00	b011	b01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	b00	b100	b00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	b00	b100	b01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
16-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
0	b01	b000	b00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	b01	b000	b01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	b01	b001	b00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	b01	b001	b01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	b01	b010	b00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	b01	b010	b01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	b01	b011	b00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	b01	b011	b01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	b01	b100	b00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	b01	b100	b01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
32-bit external bus high-performance address mapping (Row, Bank, Column)				
1	b00	b000	b00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9

Table 3-22 Address mapping (continued)

[14]	[13:12]	[11:9]	[8:7]	Description
1	b00	b000	b01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
1	b00	b001	b00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
1	b00	b001	b01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
1	b00	b001	b10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
1	b00	b010	b00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
1	b00	b010	b01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
1	b00	b010	b10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
1	b00	b011	b00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
1	b00	b011	b01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
1	b00	b011	b10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
1	b00	b100	b00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
1	b00	b100	b01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
32-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
1	b01	b000	b00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
1	b01	b000	b01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
1	b01	b001	b00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
1	b01	b001	b01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
1	b01	b001	b10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
1	b01	b010	b00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
1	b01	b010	b01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
1	b01	b010	b10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
1	b01	b011	b00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
1	b01	b011	b01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9

Table 3-22 Address mapping (continued)

[14]	[13:12]	[11:9]	[8:7]	Description
1	b01	b011	b10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
1	b01	b100	b00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
1	b01	b100	b01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10

A chip select can be connected to a single memory device. In this case the chip select data bus width is the same as the device width. Alternatively the chip select can be connected to a number of external devices. In this case the chip select data bus width is the sum of the memory device data bus widths.

For example, for a chip select connected to:

- a 32-bit wide memory device, choose a 32-bit wide address mapping
- a 16-bit wide memory device, choose a 16-bit wide address mapping
- four x 8-bit wide memory devices, choose a 32-bit wide address mapping
- two x 8-bit wide memory devices, choose a 16-bit wide address mapping.

3.3.19 Dynamic Memory RAS and CAS Delay Registers 0-3, MPMCDynamicRasCas0-3

The eight-bit, read/write, MPMCDynamicRasCas0-3 Registers enable you to program the RAS and CAS latencies for the relevant dynamic memory. These registers must only be modified during system initialization. The MPMCDynamicRasCas0-3 Registers are accessed with one wait state. Table 3-23 lists the bit assignments for the MPMCDynamicRasCas0-3 Registers.

Table 3-23 MPMCDynamicRasCas0-3 Register bit assignments

Bits	Name	Description
[31:11]	-	Reserved, read undefined, do not modify.
[10:7]	CAS	CAS latency: b0000 = reserved b0001 = half cycle b0010 = one cycle b0011 = one and a half cycles b0100 = two cycles b0101 = two and a half cycles b0110 = three cycles b0111 = three and a half cycles b1000 = four cycles b1001 = four and a half cycles b1010 = five cycles b1011 = five and a half cycles b1100 = six cycles b1101 = six and a half cycles b1110 = seven cycles b1111 = seven and a half cycles (reset value on **nPOR**). The value of the chip select 5 CAS latency field on power-on reset (**nPOR**) is determined by the **MPMCDYCS5CASDLY[3:0]** signal. This value can be overridden by software and would then reflect the last value that is written into it. This field is unaffected by AHB reset (**HRESETn**).
[6:4]	-	Reserved, read undefined, do not modify.
[3:0]	RAS	RAS latency, active to read or write delay: b0000 = reserved b0001-b1111 = n cycles[a] b0011 = 3 cycles (reset value on **nPOR**).

a. This value must be smaller than the t_{RAS} clock cycle setting, in the *Dynamic Memory Active To Precharge Command Period Register, MPMCDynamictRAS* on page 3-18

3.3.20 Static Memory Configuration Registers 0-3, MPMCStaticConfig0-3

The seven-bit, read/write, MPMCStaticConfig0-3 Registers are used to configure the static memory configuration. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power, or disabled mode.

The MPMCStaticConfig0-3 registers are accessed with one wait state.

Table 3-24 lists the bit assignments for the MPMCStaticConfig0-3 Registers.

Table 3-24 MPMCStaticConfig0-3 Register bit assignments

Bits	Name	Description
[31:21]	-	Reserved, read undefined, do not modify.
[20]	P	Write protect: 0 = writes not protected (reset value on **nPOR**) 1 = write-protected.
[19:9]	-	Reserved, read undefined, do not modify.
[8]	EW	Extended wait: 0 = Extended wait disabled (reset value on **nPOR**) 1 = Extended wait enabled. Extended wait (EW) uses the MPMCStaticExtendedWait register to time both the read and write transfers rather than the MPMCStaticWaitRd and MPMCStaticWaitWr registers. This enables much longer transactions.[a]

Table 3-24 MPMCStaticConfig0-3 Register bit assignments (continued)

Bits	Name	Description
[7]	PB	Byte lane state: 0 = For reads all four signals in **nMPMCBLSOUT[3:0]** are HIGH. For writes the respective active signals in **nMPMCBLSOUT[3:0]** are LOW (reset value for chip select 0, 2, and 3 on **nPOR**). 1 = For reads all four signals in **nMPMCBLSOUT[3:0]** are LOW. For writes the respective active signals in **nMPMCBLSOUT[3:0]** are LOW. The value of the chip select 1 byte lane state field on power-on reset (**nPOR**) is determined by the **MPMCSTCS1PB** signal. This value can be overridden by software. This field is unaffected by AHB reset (**HRESETn**). The byte lane state bit, PB, enables different types of memory to be connected. For byte-wide static memories the **nMPMCBLSOUT[3:0]** signals from the MPMC are usually connected to **nWE** (write enable). In this case for reads all the **nMPMCBLSOUT[3:0]** signals must be HIGH. This means that the byte lane state (PB) bit must be LOW. 16-bit wide static memory devices usually have the **nMPMCBLSOUT[3:0]** signals connected to the **nUB** and **nLB** (upper byte and lower byte) signals in the static memory. In this case a write to a particular byte must assert LOW the appropriate **nUB** or **nLB** signal. For reads, all the **nUB** and **nLB** signals must be asserted LOW so that the bus is driven. In this case the byte lane state (PB) bit must be HIGH.[b]
[6]	PC	Chip select polarity: 0 = active LOW chip select 1 = active HIGH chip select. The value of the chip select polarity on power-on reset (**nPOR**) is determined by the relevant **MPMCSTCSxPOL** signal. This value can be overridden by software. This field is unaffected by AHB reset (**HRESETn**).[c]
[5:4]	-	Reserved, read undefined, do not modify.

Table 3-24 MPMCStaticConfig0-3 Register bit assignments (continued)

Bits	Name	Description
[3]	PM	Page mode: 0 = disabled (reset value on **nPOR**) 1 = asynchronous page mode enabled (page length four). In page mode the MPMC can burst up to four external accesses. Therefore devices with asynchronous page mode burst four or higher devices are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.
[2]	-	Reserved, read undefined, do not modify.
[1:0]	MW	Memory width: b00 = 8-bit (reset value for chip select 0, 2, and 3 on **nPOR**) b01 = 16-bit b10 = 32-bit b11 = reserved. The value of the chip select 1 memory width field on power-on reset (**nPOR**) is determined by the **MPMCSTCS1MW[1:0]** signal. This value can be overridden by software. This field is unaffected by AHB reset (**HRESETn**).[d]

a. Extended wait and page mode cannot be selected simultaneously.
b. For chip select 1 the value of the **MPMCSTCS1PB** signal is reflected in this field. When programmed this register reflects the last value that is written into it.
c. The value of the relevant **MPMCSTCSxPOL** signal is reflected in this field. When programmed this register reflects the last value that is written into it.
d. For chip select 1 the value of the **MPMCSTCS1MW[1:0]** signal is reflected in this field. When programmed this register reflects the last value that is written into it.

Note

Synchronous burst mode memory devices are not supported.

3.3.21 Static Memory Write Enable Delay Registers 0-3, MPMCStaticWaitWen0-3

The four-bit, read/write, MPMCStaticWaitWen0-3 Registers enable you to program the delay from the chip select to the write enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode.

The MPMCStaticWaitWen0-3 Registers are accessed with one wait state.

Table 3-25 lists the bit assignments for the MPMCStaticWaitWen0-3 Registers.

Table 3-25 MPMCStaticWaitWen0-3 Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	WAITWEN	Wait write enable, delay from chip select assertion to write enable: b0000 = one **HCLK** cycle delay between assertion of chip select and write enable (reset value on **nPOR**) b0001-b1111 = (n+1) **HCLK** cycle delay[a].

a. The delay is (WAITWEN +1) x t_{HCLK}.

3.3.22 Static Memory Output Enable Delay Registers 0-3, MPMCStaticWaitOen0-3

The four-bit, read/write, MPMCStaticWaitOen0-3 Registers enable you to program the delay from the chip select or address change, whichever is later, to the output enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode.

The MPMCStaticWaitOen0-3 Registers are accessed with one wait state.

Table 3-26 lists the bit assignments for the MPMCStaticWaitOen0-3 Registers.

Table 3-26 MPMCStaticWaitOen0-3 Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	WAITOEN	Wait output enable, delay from chip select assertion to output enable: b0000 = No delay (reset value on **nPOR**) b0001-b1111= n cycle delay[a].

a. The delay is WAITOEN x t_{HCLK}.

3.3.23 Static Memory Read Delay Registers 0-3, MPMCStaticWaitRd0-3

The five-bit, read/write, MPMCStaticWaitRd0-3 Registers enable you to program the delay from the chip select to the read access. It is recommended that this register is modified during system initialization, or when there are no current or outstanding

transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. It is not used if the extended wait bit is enabled in the MPMCStaticWaitRd0-3 Registers.

The MPMCStaticWaitRd0-3 Registers are accessed with one wait state.

Table 3-27 lists the bit assignments for the MPMCStaticWaitRd0-3 Registers.

Table 3-27 MPMCStaticWaitRd0-3 Register bit assignments

Bits	Name	Description
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	WAITRD	Non page-mode read or asynchronous page mode read, first read only: b00000-b11110 = (n+1) **HCLK** cycles for read accesses[a] b11111 = 32 **HCLK** cycles for read accesses (reset value on **nPOR**).

a. For nonsequential reads, the wait state time is (WAITRD + 1) x t_{HCLK}.

3.3.24 Static Memory Page Mode Read Delay Registers 0-3, MPMCStaticWaitPage0-3

The five-bit, read/write, MPMCStaticWaitPage0-3 Registers enable you to program the delay for asynchronous page mode sequential accesses. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode.

The MPMCStaticWaitPage0-3 Registers are accessed with one wait state.

Table 3-28 lists the bit assignments for the MPMCStaticWaitPage0-3 Registers.

Table 3-28 MPMCStaticWaitPage0-3 Register bit assignments

Bits	Name	Description
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	WAITPAGE	Asynchronous page mode read after the first read wait states. Number of wait states for asynchronous page mode read accesses after the first read: b00000-b11110 = (n+ 1) **HCLK** cycle read access time[a] b11111 = 32 **HCLK** cycle read access time (reset value on **nPOR**).

a. For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE + 1) x t_{HCLK}

3.3.25 Static Memory Write Delay Registers 0-3, MPMCStaticWaitWr0-3

The five-bit, read/write, MPMCStaticWaitWr0-3 Registers enable you to program the delay from the chip select to the write access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are not used if the extended wait (EW) bit is enabled in the MPMCStaticConfig Register.

The MPMCStaticWaitWr0-3 Registers are accessed with one wait state.

Table 3-29 lists the bit assignments for the MPMCStaticWaitWr0-3 Registers.

Table 3-29 MPMCStaticWaitWr0-3 Register bit assignments

Bits	Name	Description
[31:5]	-	Reserved, read undefined, do not modify.
[4:0]	WAITWR	Write wait states, SRAM wait state time for write accesses after the first read: b00000-b11110 = (n+2) **HCLK** cycle write access time[a] b11111 = 33 **HCLK** cycle write access time (reset value on **nPOR**).

a. The wait state time for write accesses after the first read is (WAITWR +2) x t_{HCLK}

3.3.26 Static Memory Turn Round Delay Registers 0-3, MPMCStaticWaitTurn0-3

The four-bit, read/write, MPMCStaticWaitTurn0-3 Registers enable you to program the number of bus turnaround cycles. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode.

The MPMCStaticWaitTurn0-3 Registers are accessed with one wait state.

Table 3-30 lists the bit assignments for the MPMCStaticWaitTurn0-3 Registers.

Table 3-30 MPMCStaticWaitTurn0-3 Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined, do not modify.
[3:0]	WAITTURN	Bus turnaround cycles: b0000-b1110 = (n+1) **HCLK** turnaround cycles[a] b1111 = 16 **HCLK** turnaround cycles (reset value on **nPOR**).

a. Bus turnaround time is (WAITTURN + 1) x t_{HCLK}

To prevent bus contention on the external memory data bus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses.

The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses.

3.3.27 AHB Control Registers 0-4, MPMCAHBControl0-4

The MPMCAHBControl0-4 Registers are one-bit, read/write registers. They are used to control the AHB interfaces operation. The register fields can be altered during normal operation.

Table 3-31 lists the bit assignments for the MPMCAHBControl0-4 Registers.

Table 3-31 MPMCAHBControl0-4 Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved, read undefined, do not modify.
[0]	E	Buffer enable, enable AHB port buffer: 0 = disable buffer (reset value on **nPOR**) 1 = enable buffer.

The **HPROT[2]** signal is used (bufferable signals) to determine if merging takes place, and the E bit is used to determine if the AHB write buffer is used to hold the transfer. Table 3-32 lists the possible transfer types.

Table 3-32 Transfer types

E bit	HPROT [2]	Transfer
0	0	Nonbuffered, no data merging
0	1	Nonbuffered, data merging allowed
1	0	Buffered, no data merging
1	1	Buffered, data merging allowed

Note

In Table 3-32 on page 3-37:

- Buffered means **HREADY** is returned immediately for first or last transfer.
- Nonbuffered means **HREADY** is held off until transfer is placed in memory.
- Merging allowed means non word burst transfers are converted into word transfers (INCR4 BYTE is passed as 1 word write).
- No data merging means each transfer of the burst is passed through as is (INCR4 BYTE is passed as 4 byte writes).

3.3.28 AHB Status Registers 0-4, MPMCAHBStatus0-4

The MPMCAHBStatus0-4 Registers are single-bit, read-only registers. They provide AHB interface status information.

Table 3-33 lists the bit assignments for the MPMCAHBStatus0-4 Registers.

Table 3-33 MPMCAHBStatus0-4 Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved, read undefined.
[1]	S	Buffer status: 0 = buffer empty (reset value on **nPOR**) 1 = buffer contains data.
[0]	-	Reserved, read undefined.

3.3.29 AHB TimeOut Registers 0-4, MPMCAHBTimeOut0-4

The MPMCAHBTimeOut0-4 Registers are ten-bit, read/write registers. They are used to ensure that each AHB port is serviced within a programmed number of cycles. When an AHB request goes active the values in the MPMCAHBTimeOut0-4 Registers are loaded into a down counter. If the transfer is not processed by the time the TimeOut has counted down to 0, the priority of the AHB port is increased until the request is serviced.

These registers therefore enable the transaction latency, and indirectly the bandwidth, for a particular port to be defined. The value programmed into these registers depends on the latency required for the particular port.

The register fields can be altered during normal operation.

Table 3-34 lists the bit assignments for the MPMCAHBTimeOut0-4 Registers.

Table 3-34 MPMCAHBTimeOut0-4 Register bit assignments

Bits	Name	Function
[31:10]	-	Reserved, read undefined, do not modify.
[9:0]	AHBTIMEOUT	AHB time-out: `0x0` = time-out disabled (reset value on **nPOR**) 1-1023 = number of AHB clock cycles before time-out reached.

3.3.30 Additional Peripheral Identification Registers, MPMCPeriphID4-7

The MPMCPeriphID4-7 Registers are four eight-bit read-only registers, that span address locations `0xFD0-0xFDC`. The registers can conceptually be treated as a single register that holds a 32-bit additional peripheral ID value.

Table 3-35 lists the bit assignments for the conceptual 32-bit MPMC Additional Peripheral ID Register.

Table 3-35 Conceptual MPMC Additional Peripheral ID Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	Configuration1	Additional peripheral configuration information

The configuration options are peripheral-specific. For MPMC, the four, eight-bit peripheral identification registers are described in the following subsections:

- *Additional Peripheral Identification Register 4, MPMCPeriphID4*
- *Additional Peripheral Identification Registers 5-7, MPMCPeriphID5-7* on page 3-41.

Additional Peripheral Identification Register 4, MPMCPeriphID4

The MPMCPeriphID4 Register is hard-coded and the fields within the register determine the reset value. Table 3-36 lists the bit assignment of the MPMCPeriphID4 register.

Table 3-36 MPMCPeriphID4 Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved, read undefined.
[3:0]	Configuration	Number of memory ports: b0000-b1111 = n+1 memory ports b0101 = 6 memory ports (value for GX175).

Additional Peripheral Identification Registers 5-7, MPMCPeriphID5-7

The MPMCPeriphID5-7 Registers are reserved. Table 3-37 lists the bit assignments for the MPMCPeriphID5-7 Registers.

Table 3-37 MPMCPeriphID5-7 Register bit assignments

Bits	Name	Description
[31:0]	-	Reserved, read undefined

3.3.31 Peripheral Identification Registers, MPMCPeriphID0-3

The MPMCPeriphID0-3 Registers are four eight-bit read-only registers, that span address locations `0xFE0-0xFEC`. The registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value.

Table 3-38 lists the bit assignments for the conceptual 32-bit MPMC Peripheral Identification Register.

Table 3-38 Conceptual MPMC Peripheral ID Register bit assignments

Bits	Name	Description
[31:24]	Configuration	Configuration options are peripheral-specific. See *Peripheral Identification Register 3, MPMCPeriphID3 register* on page 3-44.
[23:20]	Revision	The peripheral revision number is revision dependent.
[19:12]	Designer	Designer's ID number. This is `0x41` for ARM.
[11:0]	Part number	Identifies the peripheral. The part number for GX175 is `0x175`.

Figure 3-1 on page 3-42 shows the correspondence between bits of the MPMCPeriphID0-3 registers and the conceptual 32-bit MPMC Peripheral ID register.

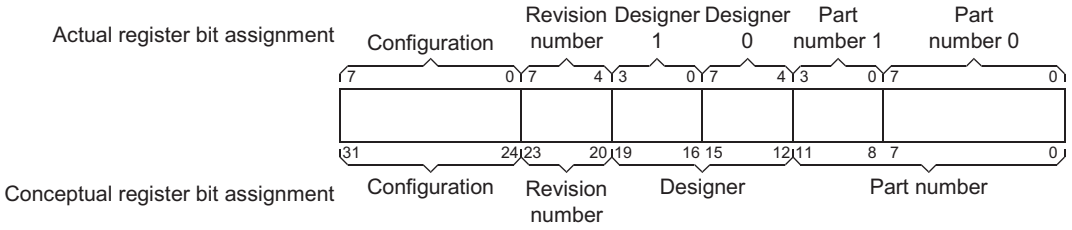


Figure 3-1 Peripheral identification register bit assignment

The four eight-bit peripheral identification registers are described in:

- *Peripheral Identification Register 0, MPMCPeriphID0*
- *Peripheral Identification Register 1, MPMCPeriphID1 register* on page 3-43
- *Peripheral Identification Register 2, MPMCPeriphID2 register* on page 3-43
- *Peripheral Identification Register 3, MPMCPeriphID3 register* on page 3-44.

Peripheral Identification Register 0, MPMCPeriphID0

The MPMCPeriphID0 Register is hard-coded and the fields within the register determine the reset value. Table 3-39 lists the bit assignments for the MPMCPeriphID0 Register.

Table 3-39 MPMCPeriphID0 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	PartNumber0	These bits read back as `0x75`

Peripheral Identification Register 1, MPMCPeriphID1 register

The MPMCPeriphID1 Register is hard-coded and the fields within the register determine the reset value. Table 3-40 lists the bit assignments for the MPMCPeriphID1 Register.

Table 3-40 MPMCPeriphID1 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:4]	Designer0	These bits read back as 0x1
[3:0]	PartNumber1	These bits read back as 0x1

Peripheral Identification Register 2, MPMCPeriphID2 register

The MPMCPeriphID2 Register is hard-coded and the fields within the register determine the reset value. Table 3-41 lists the bit assignments for the MPMCPeriphID2 Register.

Table 3-41 MPMCPeriphID2 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:4]	Revision	These bits read back as the revision number. 0x0 = GX175 revision r0p0 0x1 = GX175 revision r0p1 0x2 = GX175 revision r0p2
[3:0]	Designer1	These bits read back as 0x4

Peripheral Identification Register 3, MPMCPeriphID3 register

The MPMCPeriphID3 register is hard-coded and the fields within the register determine the reset value. Table 3-42 lists the bit assignments for the MPMCPeriphID3 register.

Table 3-42 MPMCPeriphID3 Register bit assignments

Bits	Name	Description
[31:7]	-	Reserved, read undefined.
[6]	Configuration	MBX Interface Port: 0 = no 1 = yes For GX175 this field is set to 1.
[5:3]	Configuration	Indicates the AHB master bus width: b000 = 32-bit wide b001 = 64-bit wide b010 = 128-bit wide b011 = 256-bit wide b100 = 512-bit wide b101 = 1024-bit wide b110-b111 = reserved For GX175 this field is set to b000.
[2]	Configuration	TIC interface: 0 = no 1 = yes For GX175 this field is set to 1.
[1]	Configuration	Data buffers: 0 = no 1 = yes For GX175 this field is set to 1.
[0]	Configuration	Static memory controller: 0 = no 1 = yes For GX175 this field is set to 1.

3.3.32 PrimeCell Identification Registers 0-3, MPMCPCellID0-3

The MPMCPCellID0-3 Registers are four eight-bit wide registers, that span address locations 0xFF0-0xFFC. The registers can conceptually be treated as a single register that holds a 32-bit PrimeCell ID value. The register can be used for automatic BIOS configuration. The MPMCCellID Register is set to 0xB105F00D.

The register can be accessed with one wait state.

Table 3-43 lists the bit assignments for the conceptual PrimeCell ID Register.

Table 3-43 Conceptual PrimeCell ID Register bit assignments

PrimeCell ID register		MPMCPCellID0-3 register		
Bits	**Reset value**	**Register**	**Bits**	**Description**
-	-	MPMCPCellID3	[31:8]	Reserved, read undefined
[31:24]	0xB1	MPMCPCellID3	[7:0]	These bits read back as 0xB1
-	-	MPMCPCellID2	[31:8]	Reserved, read undefined
[23:16]	0x05	MPMCPCellID2	[7:0]	These bits read back as 0x05
-	-	MPMCPCellID1	[31:8]	Reserved, read undefined
[15:8]	0xF0	MPMCPCellID1	[7:0]	These bits read back as 0xF0
-	-	MPMCPCellID0	[31:8]	Reserved, read undefined
[7:0]	0x0D	MPMCPCellID0	[7:0]	These bits read back as 0x0D

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for integration testing. It contains the following sections:

4.1 MPMC test harness overview

The additional logic for functional verification and integration vectors enables:

- capture of input signals to the block
- stimulation of the output signals.

The integration vectors provide a way of verifying that the MPMC is correctly wired into a system. This is done by separately testing three groups of signals:

AMBA signals

These are the AMBA bus signals.

Non-AMBA intra-chip signals

Non-AMBA intra-chip signals are on-chip signals that are not part of the AMBA bus, for example the interrupt request signals.

Primary inputs and outputs

The primary input and output signals are those that go off and on the chip.

4.1.1 AMBA test strategy

These signals are tested by performing various AMBA bus transactions.

4.1.2 Non-AMBA intra-chip integration test strategy

Non-AMBA intra-chip signals are on-chip signals that are not part of the AMBA bus, for example, the interrupt request signals.

Many of these signals are difficult to control and observe. Therefore PrimeCells have integration test logic to make this task easier. These test features are enabled by programming the MPMC Integration Test Control Register, MPMCITCR. The intra-chip input signal values can then be read using the MPMCITIP register. The intra-chip output signals can be controlled using the MPMCITOP register.

4.1.3 Primary I/O test strategy

The primary I/O signals are tested by performing sequences of memory transactions.

4.2 Production test

The MPMC is designed to simplify:

- insertion of scan test cells
- use of *Automatic Test Pattern Generation* (ATPG).

Scan insertion and ATPG is the recommended method of manufacturing test.

4.3 Test registers

The MPMC test registers are memory-mapped as shown in Table 4-1.

Table 4-1 Test registers memory map

Register	Offset	Type	Width	Reset value	Description
MPMCITCR	0xF00	RW	1	0x0	See *Test Control Register, MPMCITCR*
MPMCITIP0	0xF20	RW	32	0x0	See *Test Input 0 Register, MPMCITIP0* on page 4-5
MPMCITIP1	0xF24	RW	7	0x0	See *Test Input 1 Register, MPMCITIP1* on page 4-9
MPMCITOP	0xF40	RW	4	0x1	See *Test Output Register, MPMCITOP* on page 4-10

4.3.1 Test Control Register, MPMCITCR

The MPMCITCR Register is a single-bit read/write control register. The T bit in this register controls the input test multiplexors. This register must only be used in test mode. The register can be accessed with one wait state.

Table 4-2 lists the bit assignments for the MPMCITCR Register.

Table 4-2 MPMCITCR Register bit assignments

Bits	Name	Description
[31:1]	-	Reserved, read undefined, do not modify
[0]	T	Test control register: 0 = normal mode (reset value on **nPOR** or **HRESETn**) 1 = test mode

4.3.2 Test Input 0 Register, MPMCITIP0

The MPMCITIP0 Register is a 32-bit read/write register. This register must only be used in test mode. The register can be accessed with one wait state.

Table 4-3 lists the bit assignments for the MPMCITIP0 Register.

Table 4-3 MPMCITIP0 Register bit assignments

Bits	Name	Description
[31]	MPMCDYDDRPOL	DDR-SDRAM pad interface polarity signal **MPMCDYDDRPOL**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYDDRPOL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH. 1 = Set this bit if the MPMCITCR T bit is HIGH.
[30:29]	MPMCDYDDRDLY	DDR-SDRAM pad interface configuration signal **MPMCDYDDRDLY[1:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYDDRDLY[1:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[28]	MPMCBOOTDLY	Boot delay signal **MPMCBOOTDLY**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCBOOTDLY** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[27]	MPMCDYSDRPOL	SDR-SDRAM pad interface polarity signal **MPMCDYSDRPOL**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYSDRPOL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.

Table 4-3 MPMCITIP0 Register bit assignments (continued)

Bits	Name	Description
[26:25]	MPMCDYSDRDLY	SDR-SDRAM pad interface configuration signal **MPMCDYSDRDLY[1:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYSDRDLY[1:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.
[24:21]	MPMCDYCS5CASDLY	Chip select5 CAS delay signal **MPMCDYCS5CASDLY[3:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYCS5CASDLY[3:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.
[20:13]	MPMCDYCS5ADRMAP	Chip select5 address map signal **MPMCDYCS5ADRMAP[7:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYCS5ADRMAP[7:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.
[12:10]	MPMCDYCS5DEVICE	Chip select5 device signal **MPMCDYCS5DEVICE[2:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYCS5DEVICE[2:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.

Table 4-3 MPMCITIP0 Register bit assignments (continued)

Bits	Name	Description
[9]	MPMCDLLCALIACK	Value of DLL calibration acknowledge signal **MPMCDLLCALIACK**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDLLCALIACK** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[8]	MPMCSTCS1PB	Polarity of byte lane for chip select1 signal **MPMCSTCS1PB**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS1PB** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[7]	MPMCSTCS3POL	Polarity of chip select3 signal **MPMCSTCS3POL**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS3POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[6]	MPMCSTCS2POL	Polarity of chip select2 signal **MPMCSTCS2POL**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS2POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[5]	MPMCSTCS1POL	Polarity of chip select1 signal **MPMCSTCS1POL**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS1POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.

Table 4-3 MPMCITIP0 Register bit assignments (continued)

Bits	Name	Description
[4]	MPMCSTCS0POL	Polarity of chip select0 signal **MPMCSTCS0POL**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS0POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[3:2]	MPMCSTCS1MW	Chip select 1 memory width **MPMCSTCS1MW[1:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS1MW[1:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.
[1]	MPMCBIGENDIAN	Big-endian enable signal **MPMCBIGENDIAN**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCBIGENDIAN** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[0]	MPMCSREFREQ (SR)	Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSREFREQ** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.

4.3.3 Test Input 1 Register, MPMCITIP1

The MPMCITIP1 Register is a seven-bit read/write register. This register must only be used in test mode. The register can be accessed with one wait state. Table 4-4 lists the bit assignments for the MPMCITIP1 Register.

Table 4-4 MPMCITIP1 Register bit assignments

Bits	Name	Description
[31:7]	-	Reserved, read undefined, do not modify.
[6:3]	MPMCCKEINIT	Clock enable signal **MPMCCKEINIT[3:0]**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCCKEINIT[3:0]** input signals if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[2]	MPMCDQMINIT	Data mask signal **MPMCDQMINIT**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDQMINIT** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[1]	MPMCEBIBACKOFF	EBI backoff signal **MPMCEBIBACKOFF**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCEBIBACKOFF** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[0]	MPMCEBIGNT	EBI grant signal **MPMCEBIGNT**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCEBIGNT** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.

4.3.4 Test Output Register, MPMCITOP

The MPMCITOP Register is a four-bit read/write register. This register must only be used in test mode. The register can be accessed with one wait state.

Table 4-5 lists the bit assignments for the MPMCITOP Register.

Table 4-5 MPMCITOP Register bit assignments

Bits	Name	Description
[31:10]	-	Reserved, read undefined, do not modify.
[9]	MPMCDLLCALIREQ	Value of DLL calibration request signal **MPMCDLLCALIREQ**. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDLLCALIREQ** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit if the MPMCITCR T bit is HIGH 1 = Set this bit if the MPMCITCR T bit is HIGH.
[8:3]	-	Reserved, read undefined, do not modify.
[2]	MPMCEBIREQ	Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCEBIREQ** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit and **MPMCEBIREQ** output signal if the MPMCITR T bit is HIGH 1 = Set this bit and **MPMCEBIREQ** output signal if the MPMCITR T bit is HIGH.
[1]	MPMCRPVHHOUT	Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCRPVHHOUT** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit and **MPMCRPVHHOUT** signal if the MPMCITR T bit is HIGH 1 = Set this bit and **MPMCRPVHHOUT** signal if the MPMCITR T bit is HIGH.
[0]	MPMCSREFACK (SA)	Self-refresh acknowledge. Read: Read the value of this bit if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSREFACK** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this bit and **MPMCSREFACK** output signal if the MPMCITR T bit is HIGH 1 = Set this bit and **MPMCSREFACK** output signal if the MPMCITR T bit is HIGH.

Appendix A
Signal Descriptions

This appendix describes the signals that interface with the ARM MPMC controller block. It contains the following sections:

A.1 AHB register signals

Table A-1 lists the AHB register signals.

Table A-1 AHB register signal descriptions

Name	Type	Source/ destination	Description
HADDRREG[11:2]	Input	AHB master layer	The AHB address bus.
HCLK	Input	Clock source	Bus clock. This clock times all bus transfers. All signal timings are related to the rising edge of **HCLK**.
HRDATAREG[31:0]	Output	AHB master layer	Read data bus. The read data bus is used to transfer data from the MPMC to the bus master during read operations.
HREADYINREG	Input	AHB slave layer	Transfer done. When HIGH, the **HREADYINREG** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer. An alternate slave generates this signal.
HREADYOUTREG	Output	AHB master and AHB master layer	Transfer done. When HIGH, the **HREADYOUTREG** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESETn	Input	Reset controller	Reset. The bus reset signal is active LOW and is used to reset the system and the bus.
HRESPREG[1:0]	Output	AHB master layer	Transfer response. The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY, and SPLIT. The SDRAM controller can respond with either the OKAY or ERROR responses. The ERROR response is generated when the transfer size is not 32 bits wide.
HSELMPMCREG	Input	AHB decoder	Slave select. MPMC controller register select signal. This signal indicates an access to the MPMC control registers.
HSIZEREG[2:0]	Input	AHB master layer	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). Only word (32-bit) transfers are allowed (**HSIZE[2:0]** = `b010`) to access the MPMC registers. Transfer sizes other than 32 bits generate an ERROR response.

Table A-1 AHB register signal descriptions (continued)

Name	Type	Source/ destination	Description
HTRANSREG[1:0]	Input	AHB master layer	Transfer type. Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY. Only **HTRANSREG[1]** is used, indicating if a transfer is required or not.
HWDATAREG[31:0]	Input	AHB master layer	Write data bus. The write data bus is used to transfer data from the master to the bus slaves during write operations.
HWRITEREG	Input	AHB master layer	Transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.

A.2 AHB memory signals

Table A-2 lists the AHB memory signals. The signal names for each port can be found by substituting the port number, 0, 1, 2, 3, or 4 for the symbol x. Unused ports are disabled by connecting their inputs to logic 0. The MPMC does not support RETRY or SPLIT transactions.

Table A-2 AHB memory signal descriptions

Name	Type	Source/ destination	Description
HADDRx[28:0]	Input	AHB master layer	The AHB address bus.
HBURSTx[2:0]	Input	AHB master layer	Burst type. Indicates if the transfer forms part of a burst. 4, 8, and 16 beat bursts are supported and the burst can be either incrementing or wrapping.
HMASTLOCKx	Input	AHB master layer	Locked transfers. When HIGH this signal indicates that the master requires locked access to the SDRAM and no other master must be granted the bus until this signal is LOW.
HPROTx[3:2]	Input	AHB master	The protection control signals provide additional information about a bus access. The signals indicate if the transfer is: • an opcode fetch or a data access • a privileged mode access or a User mode access • a cacheable access or a noncacheable access • a bufferable access or a nonbufferable access.
HRDATAx[31:0]	Output	AHB master layer	Read data bus. The read data bus is used to transfer data from the MPMC to the bus master during read operations.
HREADYINx	Input	AHB slave layer	Transfer done. When HIGH, the **HREADYINx** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer. An alternate slave generates this signal.
HREADYOUTx	Output	AHB master layer AHB slave layer	Transfer done. When HIGH, the **HREADYOUTx** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.

Table A-2 AHB memory signal descriptions (continued)

Name	Type	Source/ destination	Description
HRESPx[1:0]	Output	AHB master layer	Transfer response. The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY, and SPLIT. The SDRAM controller can respond with either the OKAY or ERROR responses. The ERROR response is generated when: • the transfer size is greater than allowed • the memory access is a write to a write-protected region • the memory is accessed with the MPMC enable (E) bit disabled or the low-power mode (L) bit asserted. See Table 3-2 on page 3-10.
HSELMPMCxCS[7:0]	Input	AHB decoder	Slave select. MPMC select signal. These signals indicate an access to memory. Specific chip select, **HSELMPMCxCS[0]** indicates the transfer is to chip select 0, **HSELMPMCxCS[1]** indicates the transfer is to chip select 1, and so on. Only one signal can go active at a time. The **HSELMPMCxCS** to chip select decoding is as follows: • **HSELMPMCxCS[0]** selects **nMPMCSTCSOUT[0]** • **HSELMPMCxCS[1]** selects **nMPMCSTCSOUT[1]** • **HSELMPMCxCS[2]** selects **nMPMCSTCSOUT[2]** • **HSELMPMCxCS[3]** selects **nMPMCSTCSOUT[3]** • **HSELMPMCxCS[4]** selects **nMPMCDYCSOUT[0]** • **HSELMPMCxCS[5]** selects **nMPMCDYCSOUT[1]** • **HSELMPMCxCS[6]** selects **nMPMCDYCSOUT[2]** • **HSELMPMCxCS[7]** selects **nMPMCDYCSOUT[3]**.
HSELMPMCxG	Input	AHB decoder	Slave select. MPMC global select signal. This signal must go active with the **HSELMPMCxCS[7:0]** signals, and whenever the MPMC is accessed.
HSIZEx[2:0]	Input	AHB master layer	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit), or word (32-bit). Byte (8-bit), halfword (16-bit), and word (32-bit) transfers are allowed to access the external memory. Transfer sizes greater than 32 bits generate an ERROR response.

Table A-2 AHB memory signal descriptions (continued)

Name	Type	Source/ destination	Description
HTRANSx[1:0]	Input	AHB master layer	Transfer type. Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY.
HWDATAx[31:0]	Input	AHB master layer	Write data bus. The write data bus is used to transfer data from the master to the bus slaves during write operations.
HWRITEx	Input	AHB master layer	Transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.

A.3 MBX interface port signals

The MPMC MBX interface port is a slave port that connects directly to the MBX 3D Graphics Core master port. Table A-3 lists the MBX interface port signals.

Table A-3 MBX interface port signals

Name	Type	Source/ destination	Description
GADDR[28:2]	Input	MBX 3D Graphics Core	This is the MBX 3D Graphics Core memory interface address bus for reads and writes. It is 30 bits wide, and is a 32-bit aligned (word) address. The value on this bus is captured by the MPMC when **GTRANS** = 1 and **GAREADY** = 1. If the system does not contain an MMU then the top seven bits are not used, and are tied LOW.
GAREADY	Output	MBX 3D Graphics Core	This signal is driven HIGH when the MPMC is ready to receive another transfer from the MBX 3D Graphics Core memory interface controller.
GBSTRB[3:0]	Input	MBX 3D Graphics Core	This is a four-bit signal which is a byte mask. A HIGH on the relevant byte mask indicates that the byte is valid. There are certain requestors in the MBX 3D Graphics Core that do read-modify-writes and they have a byte granularity. That is, only one or two bytes are valid and therefore the data in the remaining bytes of a 32-bit SDRAM memory location must be retained. These bits relate directly to the **DQM** signals found on SDRAM devices. The bits of **GBSTRB** relate to the bytes of **GWDATA** as follows: 0 = [7:0] 1 = [15:8] 2 = [23:16] 3 = [31:24].
GDREADY	Output	MBX 3D Graphics Core	This signal is driven HIGH by the MPMC to indicate that the returned data is valid.
GRDATA[31:0]	Output	MBX 3D Graphics Core	The read data bus is used to transfer data from the MPMC to the MBX 3D Graphics Core during read operations. The value on this bus is captured by the MBX 3D Graphics Core when **GDREADY** = 1.
GSELCS[7:4]	Input	GXI Decoder	This is the MPMC slave select signal. These signals indicate an access to a particular dynamic memory chip select.
GSELG	Input	GXI Decoder	This is the MPMC global slave select signal. This signal indicates an access to memory.

Table A-3 MBX interface port signals (continued)

Name	Type	Source/ destination	Description
GTRANS	Input	MBX 3D Graphics Core	This signal, when HIGH, indicates that the current transfer is valid and that the control signals can be sampled by the slave.
GWDATA[31:0]	Input	MBX 3D Graphics Core	The write data bus is used to transfer data from the MBX 3D Graphics Core to the MPMC during write operations. The value on this bus is captured by the MPMC when **GTRANS** = 1 and **GAREADY** = 1.
GWRITE	Input	MBX 3D Graphics Core	This signal defines whether the current transfer is a read or a write. When LOW it signifies a read. When HIGH it signifies a write. The value of this signal is captured by the MPMC when **GTRANS** = 1 and **GAREADY** = 1.

A.4 Miscellaneous signals

The MPMC miscellaneous signals are described in:

- *Miscellaneous signals*
- *Tie-off signals* on page A-10
- *Test signals* on page A-15
- *Clock signals* on page A-16
- *External Bus Interface signals* on page A-16.

A.4.1 Miscellaneous signals

Table A-4 lists the miscellaneous signals.

Table A-4 Miscellaneous signals

Name	Type	Source/ destination	Description
MPMCDLLCALIACK	Input	DLL block	DLL calibration acknowledge.
MPMCDLLCALIREQ	Output	DLL block	DLL calibration request.
MPMCSREFACK	Output	PMU	Self-refresh acknowledgement.
MPMCSREFREQ	Input	PMU	Self-refresh request.
nPOR	Input	Reset controller	Power-on reset (active LOW).
MPMCCKEINIT[3:0]	Input	Reset controller	Defines the power-on level for the **MPMCCKEOUT** signals after **nPOR** reset. When SDR/DDR memory is in self-refresh mode this value must be `0x0`. For uninitialized SDR/DDR memory this value must be `0xF`.
MPMCDQMINIT	Input	Reset controller	Defines the power-on level for the **MPMCDQMOUT** signals after **nPOR** reset. When SDR/DDR memory is in self-refresh mode this value must be 0. For uninitialized SDR/DDR memory this value must be 1.

A.4.2 Tie-off signals

Table A-5 lists the tie-off signals.

Table A-5 Tie-off signal descriptions

Name	Type	Source/ destination	Description
MPMCBIGENDIAN	Input	Reset controller	Endian select. If this bit is HIGH, big-endian mode is selected. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCBOOTDLY	Input	Reset controller	When HIGH, indicates that no accesses must be performed. Used for dynamic memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCDYCS5ADRMAP[7:0]	Input	Reset controller	Indicates the address map for chip select 5. Used for dynamic memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3* on page 3-25.
MPMCDYCS5CASDLY[3:0]	Input	Reset controller	Indicates the upper four bits of CAS delay for chip select 5. Used for dynamic memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory RAS and CAS Delay Registers 0-3, MPMCDynamicRasCas0-3* on page 3-30.

Table A-5 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCDYCS5DEVICE[2:0]	Input	Reset controller	Indicates the type of device connected to chip select 5. Used for dynamic memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Configuration Registers 0-3, MPMCDynamicConfig0-3* on page 3-25.
MPMCDYDDRDLY[1:0]	Input	Reset controller	DDR clocking scheme for data capture. Used for dynamic memory devices. The value can be overridden by writing to the DRD field of the MPMCDynamicReadConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Read Configuration Register, MPMCDynamicReadConfig* on page 3-16.
MPMCDYDDRPOL	Input	Reset controller	The polarity of the register in which the DDR read data is first captured. Used for dynamic memory devices. The value can be overridden by writing to the DRP bit of the MPMCDynamicReadConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Read Configuration Register, MPMCDynamicReadConfig* on page 3-16.

Table A-5 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCDYSDRDLY[1:0]	Input	Reset controller	SDR clocking scheme for data capture. Used for dynamic memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. The value can be overridden by writing to the SRD field of the MPMCDynamicReadConfig Register. For more information on the tie-off values see *Dynamic Memory Read Configuration Register, MPMCDynamicReadConfig* on page 3-16.
MPMCDYSDRPOL	Input	Reset controller	The polarity of the flip-flop in which the SDR read data is first captured. Used for dynamic memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value. The value can be overridden by writing to the SRP bit of the MPMCDynamicReadConfig Register. For more information on the tie-off values see *Dynamic Memory Read Configuration Register, MPMCDynamicReadConfig* on page 3-16.
MPMCSTCS0POL	Input	Reset controller	Chip select 0 polarity select. A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.

Table A-5 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCSTCS1MW[1:0]	Input	Reset controller	Chip select 1 memory width selection: 00 = 8-bit wide memory 01 = 16-bit wide memory 10 = 32-bit wide memory 11 = reserved. Used for static memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS1PB	Input	Reset controller	Chip select 1 byte lane polarity select. 0 = for reads all four signals in **nMPMCBLSOUT[3:0]** are HIGH. For writes the respective active signals in **nMPMCBLSOUT[3:0]** are LOW. 1 = for reads all four signals in **nMPMCBLSOUT[3:0]** are LOW. For writes the respective active signals in **nMPMCBLSOUT[3:0]** are LOW. Used for static memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.

Table A-5 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCSTCS1POL	Input	Reset controller	Chip select 1 polarity select on power-on reset (**nPOR**). A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS2POL	Input	Reset controller	Chip select 2 polarity select on power-on reset (**nPOR**). A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS3POL	Input	Reset controller	Chip select 3 polarity select on power-on reset (**nPOR**). A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on power-on reset (**nPOR**) in the reset controller. Alternatively, this signal can be tied to the appropriate value.

A.4.3 Test signals

Table A-6 lists the test signals.

Table A-6 Test signal descriptions

Name	Type	Source/ destination	Description
MPMCTESTACK	Output	Pad	This signal is used as test acknowledge, **MPMCTESTACK**, during TIC test mode. The test bus acknowledge signal gives external indication that the test bus has been granted and also indicates when a test access has completed. When **MPMCTESTACK** is LOW the current test vector must be extended until **MPMCTESTACK** becomes HIGH.
MPMCTESTIN	Input	Pad	Test Mode. This is pin is used to place the MPMC into TIC test mode. Used for TIC test.
MPMCTESTREQA	Input	Pad	Test bus request A. This pin is used in TIC test mode. In test mode this pin is the test bus request A input signal and is required as a dedicated device pin. During normal system operation the **MPMCTESTREQA** signal is used to request entry into the test mode. During test **MPMCTESTREQA** is used, in combination with **MPMCTESTREQB**, to indicate the type of test vector that is applied in the following cycle.
MPMCTESTREQB	Input	Pad	Test bus request B. This pin is used in TIC test mode. During test this signal is used, in combination with **MPMCTESTREQA**, to indicate the type of test vector that is applied in the following cycle. The test bus acknowledge signal gives external indication that the test bus has been granted and also indicates when a test access has completed. When **MPMCTESTACK** is LOW, the current test vector must be extended until **MPMCTESTACK** becomes HIGH.

A.4.4 Clock signals

Table A-7 lists the clock signals.

Table A-7 Clock signal descriptions

Name	Type	Source/ destination	Description
HCLKDELAY	Input	Clock source	Delayed version of **HCLK** used in command delayed mode to ensure that SDRAM set up and hold requirements are met.
HCLKx2	Input	Clock source	Double frequency clock that is synchronous to **HCLK**. This clock is required for DDR-SDRAM devices.
nHCLK	Input	Clock source	Inverted **HCLK** signal that can be used to capture the read data from SDRAM.
nHCLKx2	Input	Clock source	Inverted double frequency clock that is synchronous to **HCLK**. This clock is required for DDR-SDRAM devices.

A.4.5 External Bus Interface signals

Table A-8 lists the *External Bus Interface* (EBI) signals.

Table A-8 EBI signal descriptions

Name	Type	Source/ destination	Description
MPMCEBIBACKOFF	Input	EBI	The EBI backoff signal goes active HIGH when the EBI wants to remove the memory controller from the memory bus so that another memory controller can be granted the memory bus.
MPMCEBIGNT	Input	EBI	The EBI grant signal goes active HIGH when the EBI grants the external memory bus.
MPMCEBIREQ	Output	EBI	The EBI request signal goes active HIGH when the memory controller requires the memory bus.

A.5 Pad interface and control signals

The pad interface and control signals are described in:

- *Signal descriptions*
- *Values during reset and self-refresh* on page A-19.

A.5.1 Signal descriptions

Table A-9 lists the pad interface and control signals.

Table A-9 Pad interface and control signal descriptions

Name	Type	Source/ destination	Description
MPMCADDROUT[27:0]	Output	Pad	Address output. Used for both static and SDRAM devices. SDRAM memories use bits [14:0]. Static memories use bits [25:0].
MPMCCKEOUT[3:0]	Output	Pad	SDRAM clock enables. Used for SDRAM devices.
MPMCCLKOUT[3:0]	Output	Pad	SDRAM clock out. Used for SDRAM devices.
MPMCDATAIN[31:0]	Input	Pad	Read data from memory. Used for the static memory controller, the dynamic memory controller and the TIC.
MPMCDATAOUT[31:0]	Output	Pad	Data output to memory. Used for the static memory controller, the dynamic memory controller and the TIC.
MPMCDQMOUT[3:0]	Output	Pad	Data mask output to SDRAMs. Used for SDRAM devices.
MPMCDQSIN[1:0]	Input	Pad	Data strobe in. This signal is used to capture read data on the positive edge of the data strobe. Used for DDR-SDRAM devices.
MPMCDQSOUT[1:0]	Output	Pad	Data strobe out. Used for DDR-SDRAM.
MPMCFBCLKIN[3:0]	Input	Pad	SDRAM feedback clock in. Used for SDRAM devices.
MPMCnDQSIN[1:0]	Input	Pad	Data strobe in. This signal is the inverted data strobe signal. It is used to capture read data on the negative edge of the data strobe. Used for DDR-SDRAM devices.
MPMCRPVHHOUT	Output	Pad	Voltage control for Micron SyncFlash reset signal (RP).
nMPMCBLSOUT[3:0]	Output	Pad	Byte lane select, active LOW, for static memories. Used for static memory devices.
nMPMCCASOUT	Output	Pad	Column address strobe. Used for SDRAM devices.

Table A-9 Pad interface and control signal descriptions (continued)

Name	Type	Source/ destination	Description
nMPMCCLKOUT[3:0]	Output	Pad	DDR-SDRAM clocks. Used for DDR-SDRAM devices.
nMPMCDATAEN[3:0]	Output	Pad control	Tristate I/O pad write enable for the byte lanes of an external memory I/O data bus, for example **MPMCDATA[31:0]**, and is active LOW. Enables the byte lanes [31:24], [23:16], [15:8], and [7:0] of the data bus independently. Used for the static and dynamic memory controllers, and the TIC.
nMPMCDQSOUTEN[1:0]	Output	Pad	Tristate I/O pad output enable for the data strobe signals, **MPMCDQSIN[1:0]** and **MPMCDQSOUT[1:0]** active LOW. Used for DDR-SDRAM devices.
nMPMCDYCSOUT[3:0]	Output	Pad	SDRAM chip selects. Used for SDRAM devices.
nMPMCOEOUT	Output	Pad	Output enable for static memories. Used for static memory devices.
nMPMCRASOUT	Output	Pad	Row address strobe. Used for SDRAM devices.
nMPMCRPOUT	Output	Pad	Reset power down to SyncFlash memory. Used for the dynamic memory controller.
nMPMCSTCSOUT[3:0]	Output	Pad	Static memory chip selects. Default active LOW. Used for static memory devices.
nMPMCWEOUT	Output	Pad	Write enable. During test mode acts as test acknowledge signal. Used for SDRAM and static memories.

A.5.2 Values during reset and self-refresh

Table A-10 lists the pad interface and control signal values during reset and self-refresh.

Table A-10 Values during reset and self-refresh

Name	Value on reset (nPOR)	Value during self-refresh
MPMCADDROUT[27:0]	0x0000000	Depends on static memory accesses
MPMCCKEOUT[3:0]	Depends on **MPMCCKEINIT[3:0]**	0x0
MPMCCLKOUT[3:0]	Follows **HCLK**	Follows **HCLK**
MPMCDATAIN[31:0]	-	-
MPMCDATAOUT[31:0]	0x00000000	Depends on static memory accesses
MPMCDQMOUT[3:0]	Depends on **MPMCDQMINIT**	0x0
MPMCDQSIN[1:0]	-	-
MPMCDQSOUT[1:0]	0x0	0x0
MPMCFBCLKIN[3:0]	-	-
MPMCnDQSIN[1:0]	-	-
MPMCRPVHHOUT	0	-
nMPMCBLSOUT[3:0]	0xF	Depends on static memory accesses
nMPMCCASOUT	1	1
nMPMCCLKOUT[3:0]	Follows **nHCLK**	Follows **nHCLK**
nMPMCDATAEN[3:0]	0xF	0xF
nMPMCDQSOUTEN[1:0]	0x3	0x3
nMPMCDYCSOUT[3:0]	0xF	0xF
nMPMCOEOUT	1	Depends on static memory accesses
nMPMCRASOUT	1	1
nMPMCRPOUT	0	-
nMPMCSTCSOUT[3:0]	0xF	Depends on static memory accesses
nMPMCWEOUT	1	Depends on static memory accesses

A.6 Test Interface Controller (TIC) AHB signals

Table A-11 lists the TIC signals.

Table A-11 TIC signal descriptions

Name	Type	Source/ destination	Description
HADDRTIC[31:0]	Output	AHB slave layer	Address bus. The 32-bit AHB address bus.
HBUSREQTIC	Output	AHB arbiter	Bus request. A signal from the TIC to the bus arbiter indicates that it requires the bus.
HBURSTTIC[2:0]	Output	AHB slave layer	Burst type. Indicates if the transfer forms part of a burst. The TIC always performs incrementing bursts of unspecified length.
HGRANTTIC	Input	AHB arbiter	Bus grant. This signal indicates that the TIC is currently the highest priority master. Ownership of the address and control signals changes at the end of a transfer when **HREADYINTIC** is HIGH, so the TIC gains access to the bus when both **HREADYINTIC** and **HGRANTTIC** are HIGH.
HLOCKTIC	Output	AHB arbiter	Locked transfers. When HIGH this signal indicates that the TIC requires locked access to the bus. No other master must be granted the bus until this signal is LOW.
HPROTTIC[3:0]	Output	AHB slave layer	Protection control. The protection control signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a supervisor mode access or a User mode access. These signals also indicate if the current access is cacheable or bufferable.
HRDATATIC[31:0]	Input	AHB master layer	Read data bus. The read data bus is used to transfer data from the AHB slave to the TIC during read operations.
HREADYINTIC	Input	AHB slave layer	Transfer done. When HIGH the **HREADYINTIC** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESPTIC[1:0]	Input	AHB slave layer	Transfer response. The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY, and SPLIT.
HSIZETIC[2:0]	Output	AHB slave layer	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit), or word (32-bit). The TIC does not support larger transfer sizes.

Table A-11 TIC signal descriptions (continued)

Name	Type	Source/ destination	Description
HTRANSTIC[1:0]	Output	AHB slave layer	Transfer type. Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY. The TIC does not use the BUSY transfer type.
HWDATATIC[31:0]	Output	AHB slave layer	Write data bus. The write data bus is used to transfer data from the master to the bus slaves during write operations.
HWRITETIC	Output	AHB slave layer	Transfer direction. When HIGH, this signal indicates a write transfer and when LOW a read transfer.

A.7 Scan test signals

Table A-12 lists the scan test signals.

Table A-12 Scan test signal descriptions

Name	Type	Source/ destination	Description
SCANENABLE	Input	Test controller	Scan enable
SCANINHCLK	Input	Test controller	Scan data input for **HCLK** scan chain
SCANINHCLKDELAY	Input	Test controller	Scan in for **HCLKDELAY** scan chain
SCANINHCLKx2	Input	Test controller	Scan data input for **HCLKx2** scan chain
SCANINMPMCDQSIN[1:0]	Input	Test controller	Scan in for **MPMCDQSIN[1:0]** scan chain
SCANINFBCLKIN[3:0]	Input	Test controller	Scan in for **MPMCFBCLKIN[3:0]** scan chains
SCANINMPMCnDQSIN[1:0]	Input	Test controller	Scan in for **MPMCnDQSIN[1:0]** scan chain
SCANINnHCLK	Input	Test controller	Scan data input for **nHCLK** scan chain
SCANINnHCLKx2	Input	Test controller	Scan data input for **nHCLKx2** scan chain
SCANOUTHCLK	Output	Test controller	Scan data output for **HCLK** scan chain
SCANOUTHCLKDELAY	Output	Test controller	Scan data output for **HCLKDELAY** scan chain
SCANOUTHCLKx2	Output	Test controller	Scan data output for **HCLKx2** scan chain
SCANOUTDQSIN{1:0]	Output	Test controller	Scan out for **MPMCDQSIN{1:0]** scan chain
SCANOUTFBCLKIN[3:0]	Output	Test controller	Scan out for **MPMCFBCLKIN[3:0]** scan chain
SCANOUTnDQSIN{1:0]	Output	Test controller	Scan out for **MPMCnDQSIN{1:0]** scan chain
SCANOUTnHCLK	Output	Test controller	Scan data output for **nHCLK** scan chain
SCANOUTnHCLKx2	Output	Test controller	Scan data output for **nHCLKx2** scan chain