L210 Cache Controller

Revision r0p5

Technical Reference Manual

ARM®

ARM DDI 0284G

L210 Cache Controller
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
May 2003	A	Limited Confidential	For Limited release
June 2003	B	NonConfidential	Final, r0p1, latency signals added, Appendix E added
October 2003	C	NonConfidential	R0p2. Product name corrected, Configurability of master and slave ports clarified, Figure 4-5, L210 Miss, redrawn, Section 5.2 rewritten, LRB1 replaces LRB 0 as MBIST, **HMASTERMx** signal descriptions corrected, Appendix E extended to cover ARM 7 cores and ARM 9 cores Glossary updated, extended and corrected.
23 April 2004	D	NonConfidential	R0p3 updates. Minor modifications to register map and cache maintenance tables. Additional comments to Cache Sync maintenance operation. Preventing or reducing cache pollution section rewritten. Sections on L210 Clocking, Idle, disabled states moved to chapter 4. Comments added to section 4.6.1 concerning additional piece of hardware provided. Additional comments to unaligned burst accesses and known length bursts in section 4.6.2. Default value for **HMASTERM0** removed. Additional comments for **HMASTERM1, HMASTERM2**, **HPROTMx**, **HSIDEBANDx** signals. Additional appendix section D3 Mapping of slave to master port bursts. Section E.2.1 program listing corrected. Section E.2.2 only relevant to ARM926 revisions r0p0-r0p4, fixed from r0p5
27 May 2005	E	NonConfidential	R0p4 updates. Section 1.5 corrected. Section 2.2 Introductory text to Table 2-1 clarified. Table 2-4 added. Section 2.3.5 Description of background operations clarified.
19 September 2005	F	NonConfidential	R0p5 updates. Table 2-4 updated. Section 4.1.4 Cache Sync rewritten.
20 September 2006	G	NonConfidential	Event Monitor text clarified. Product renamed to become ARM PrimeCell L210 Level 2 Cache Controller. Section 3.2 removed. Table C-5 on page C-7 rewritten for clarity. Appendix E replaced by Applications Note *Using the L210 Cache Controller with ARM7 and ARM9 cores* (ARM DAI 0169A) Table A-10 on page A-13 footnote reworded.

Proprietary Notice

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

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

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

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

Where the term ARM is used it means “ARM or any of its subsidiaries as appropriate”

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Contents
L210 Cache Controller Technical Reference Manual

Preface
About this manual xii
Feedback xvii

Chapter 1 Introduction
1.1 About the Cache Controller 1-2
1.2 Cache controller block diagram 1-3
1.3 Functional description 1-5
1.4 Supported ARM architectures 1-10
1.5 L210 Cache Controller product revisions 1-11

Chapter 2 Programmer's Model
2.1 ARM register fields 2-2
2.2 About the cache controller registers 2-3
2.3 Register summary 2-4
2.4 Replacement strategy 2-19
2.5 Register 15: Test and Debug 2-20
2.6 Buffers 2-23
2.7 Ports configuration 2-25
2.8 Hazards 2-27

2.9 External abort support for L3 memory 2-28

Chapter 3 RAM Interfaces
3.1 About RAM interfaces 3-2
3.2 RAM configuration versus associativity and way size 3-3
3.3 Data RAM 3-7
3.4 Tag RAM 3-9
3.5 Dirty RAM 3-11

Chapter 4 Using the Cache Controller
4.1 Configuring the cache controller 4-2
4.2 Clocking 4-6
4.3 Idle 4-7
4.4 Disabled 4-8
4.5 Set-up sequence 4-9
4.6 Behavior for ARMv5 memory systems 4-10
4.7 Behavior for ARMv6 memory systems 4-14
4.8 Timing diagrams 4-20

Chapter 5 Design for Test
5.1 About design for test 5-2
5.2 Memory built-in self-test, MBIST 5-3

Chapter 6 Parity and RAM Error Support
6.1 Parity and RAM error support 6-2
6.2 Error signaling and handling 6-4

Appendix A Signal Descriptions
A.1 Cache controller signals A-2

Appendix B AC Parameters
B.1 Cache controller interface signal timing parameters B-2

Appendix C Event Monitor
C.1 Cache controller event monitor C-2
C.2 Programmer's model C-4
C.3 Implementation details C-9
C.4 Event monitor signals C-12

Appendix D Master and Slave Port Configurations
D.1 ARM1136 memory system configurations D-2
D.2 ARM926EJ-S and ARM1026EJ-S memory systems D-4
D.3 Mapping of slave to master port bursts D-6

Glossary

List of Tables
L210 Cache Controller Technical Reference Manual

	Change History	ii
Table 1-1	Typical memory sizes and access times	1-2
Table 1-2	Transactions for a three-master port system	1-8
Table 1-3	Transactions for a two-master port system	1-9
Table 1-4	Transactions for a single-master port system	1-9
Table 2-1	Summary register map	2-3
Table 2-2	Register map	2-4
Table 2-3	ID Register encoding	2-7
Table 2-4	Release number index values and releases	2-7
Table 2-5	Cache Type Register	2-8
Table 2-6	Control Register	2-9
Table 2-7	Auxiliary Control Register	2-10
Table 2-8	Cache maintenance operations	2-12
Table 2-9	Writing or reading register to way bits	2-15
Table 2-10	Line Tag Register mapping	2-21
Table 2-11	Debug Control Register	2-21
Table 3-1	Cache associativity and bits[15:13] of data RAM bus address	3-3
Table 3-2	Way size and bits[12:0] of data RAM bus address	3-4
Table 3-3	Tag RAM and way size	3-4
Table 3-4	Way size and TAGADDR values	3-5
Table 3-5	Associativity and DIRTYRD and DIRTYWD values	3-6

Table 3-6	Values for N in RAM size diagrams	3-7
Table 4-1	Total cache size compared to way size and associativity	4-2
Table 4-2	Asynchronous mode control pins	4-4
Table 4-3	HPROTSx[4:2] and TLB correspondences in an ARMv5 system	4-11
Table 4-4	Behavior for ARMv5 transactions	4-12
Table 4-5	HPROTSx[4-2] and TLB correspondences in ARMv6	4-14
Table 4-6	Behavior with ARMv6 memory types	4-15
Table 4-7	Example uses of HBSTRBSx and HUNALIGNSx, little-endian	4-18
Table 5-1	Using MBISTADDR as an index for data RAM writes	5-5
Table 5-2	MBISTADDR and MBISTDIN mappings for data RAM	5-6
Table 5-3	MBISTADDR and MBISTDIN mappings for data parity RAM	5-7
Table 5-4	MBISTADDR and MBISTDIN mapping for Tag RAMs	5-9
Table 5-5	MBISTADDR and MBISTIDIN mappings for dirty RAM	5-10
Table A-1	Slave port 0 signals	A-2
Table A-2	Master port 0 signals	A-3
Table A-3	Slave port 1 signals	A-5
Table A-4	Master port 1 signals	A-6
Table A-5	Slave port 2 signals	A-8
Table A-6	Master port 2 signals	A-9
Table A-7	Data RAM interface signals	A-11
Table A-8	Tag RAM interface signals	A-12
Table A-9	Dirty RAM interface signals	A-13
Table A-10	Cache controller event bus output signals	A-13
Table A-11	Cache controller MBIST block inputs and outputs	A-15
Table A-12	Miscellaneous signals	A-16
Table C-1	Event monitor registers	C-4
Table C-2	EMMC bit fields descriptions	C-5
Table C-3	Counter Status Register	C-6
Table C-4	Counter Configuration Register	C-6
Table C-5	Event sources encodings	C-7
Table C-6	AHB-Lite slave port interface signals	C-11
Table C-7	Event monitor input signals	C-12
Table D-1	WRAPn read access conversions on slave ports	D-6

List of Figures
L210 Cache Controller Technical Reference Manual

	Key to timing diagram conventions	xiv
Figure 1-1	Cache controller internal data paths	1-3
Figure 2-1	ID Register	2-7
Figure 2-2	Cache Type Register	2-8
Figure 2-3	Control Register	2-9
Figure 2-4	Auxiliary Control Register format	2-10
Figure 2-5	Physical address format	2-14
Figure 2-6	Index Way combination format	2-14
Figure 2-7	Lockdown format C	2-15
Figure 2-8	Test operation format	2-20
Figure 2-9	Word order in cache lines	2-20
Figure 2-10	Line Tag Register format	2-20
Figure 3-1	Data RAM bus address format	3-3
Figure 3-2	Data RAM signals	3-7
Figure 3-3	Data RAM ways	3-8
Figure 3-4	Tag RAM signals	3-9
Figure 3-5	Tag RAM organization	3-10
Figure 3-6	Dirty RAM signals	3-11
Figure 3-7	Dirty RAM memory organization	3-12
Figure 4-1	Compiled RAM latency	4-3
Figure 4-2	Asynchronous interface, read data	4-5

Figure 4-3 Asynchronous interface, write data 4-5
Figure 4-4 Hit 4-20
Figure 4-5 Miss 4-21
Figure 4-6 Simultaneous hits 4-22
Figure 4-7 Buffered write timings 4-23
Figure 4-8 Outer noncacheable access 4-24
Figure 5-1 MBIST configuration 5-4
Figure 5-2 MBIST Data RAM test paths 5-7
Figure 5-3 Data parity RAM for test 5-8
Figure 5-4 MBIST paths for tag RAM testing 5-10
Figure 5-5 Dirty RAM testing paths using MBIST 5-11
Figure 6-1 Parity and RAM error support 6-3
Figure B-1 Target timing parameters for unregistered signals B-3
Figure C-1 HCLKEN usage for a read C-10
Figure D-1 ARM1136 memory system with cache controller master port configurations D-2
Figure D-2 ARM926EJ-S and ARM1026EJ-S memory system with cache controller master port configurations D-5

Preface

This preface introduces the *ARM PrimeCell L210 Level 2 Cache Controller Technical Reference Manual*. It contains the following sections:

- *About this manual* on page xii
- *Feedback* on page xvii.

About this manual

This is the *Technical Reference Manual* (TRM) for the *ARM PrimeCell L210 Level 2 Cache Controller* (L210). For purposes of this manual the cache controller refers to the L210 Cache Controller.

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 L210 Cache Controller. The manual describes the external functionality of the cache controller.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the functionality of the cache controller.

Chapter 2 *Programmer's Model*

Read this chapter for a description of the cache controller registers and for programming details.

Chapter 3 *RAM Interfaces*

Read this chapter for details of the RAM interfaces.

Chapter 4 *Using the Cache Controller*

Read this chapter for details of how to use the cache controller effectively.

Chapter 5 *Design for Test*

Read this chapter for details of the L210 features that help with design for test issues.

Chapter 6 ***Parity and RAM Error Support***

Read this chapter for details of the parity and RAM error support.

Appendix A ***Signal Descriptions***

Read this appendix for a description of the signals used in the cache controller.

Appendix B ***AC Parameters***

Read this appendix for a description of the AC timing parameters of the cache controller signals.

Appendix C ***Event Monitor***

Read this appendix for a description of the cache controller event monitor block.

Appendix D ***Master and Slave Port Configurations***

Read this appendix for a description of the different master and slave port configurations of the cache controller.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams* on page xiv
- *Signals* on page xiv
- *Numbering* on page xv.

Typographical

The typographical conventions are:

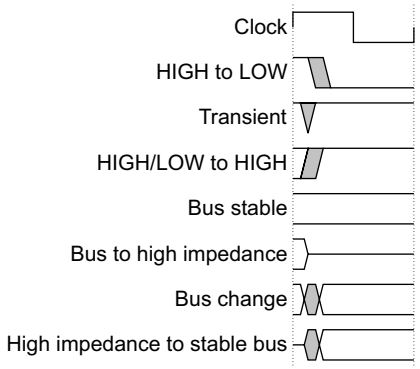
italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at 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* 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, and by third parties.

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

ARM publications

This manual contains information that is specific to the L210. See the following documents for other relevant information:

- *ARM L210 Implementation Guide (*ARM DII 0069)
- *ARM L210 MBIST Controller Technical Reference Manual* (ARM DDI 0302)

- *ARM1136J-S™ Technical Reference Manual* (ARM DDI 0211)
- *ARM1026EJ-S Technical Reference Manual* (ARM DDI 0244)
- *ARM926EJ-S PWP VC Technical Reference Manual* (ARM DDI 0232)
- *ARM720T Technical Reference Manual* (ARM DDI 0229B)
- *ARM11 AMBA (Rev 2.0) AHB Extensions* (ARM IHI 0023)
- *AMBA® Specification (Rev 2.0)* (ARM IHI 0011)
- *ARM Architecture Reference Manual (*ARM DDI 0100).
- *Using the L210 Cache Controller with ARM7 and ARM9 cores* (ARM DAI 0169)

Feedback

ARM Limited welcomes feedback on the L210 Cache Controller 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 describes the cache controller and its features. It contains the following sections:

- *About the Cache Controller* on page 1-2
- *Cache controller block diagram* on page 1-3
- *Functional description* on page 1-5
- *Supported ARM architectures* on page 1-10
- *L210 Cache Controller product revisions* on page 1-11.

1.1 About the Cache Controller

The addition of an on-chip secondary cache (also referred to as a Level 2 cache, L2CC) is a recognized method of improving the performance of computer systems when significant memory traffic is generated by the *Central Processing Unit* (CPU). By definition a secondary cache assumes the presence of a Level 1 or primary cache, closely coupled or internal to the CPU.

Memory access is fastest to L1 cache, followed closely by the L2 cache controller. Memory access is significantly slower with L3 memory, main memory. Table 1-1 shows typical sizes and access times for different types of memory.

Table 1-1 Typical memory sizes and access times

Memory type	Size	Access time
Processor registers	128 Bytes	1 cycle
On-chip L1 cache	32KB	1-2 cycles
On-chip L210 Cache Controller	128KB	8 cycles
Main memory (L3) dynamic RAM	MB GB [a]	16 cycles
Back-up memory (hard disk) (L4)	MB GB	>500 cycles

a. Size limited by the processor core addressing. For example, a 32-bit core without memory management can directly address 4GB of memory.

1.2 Cache controller block diagram

Figure 1-1 shows the internal data paths of the cache controller. It can be used in a system with an ARM1136 core, ARM1026 core, or ARM926EJ-S core.

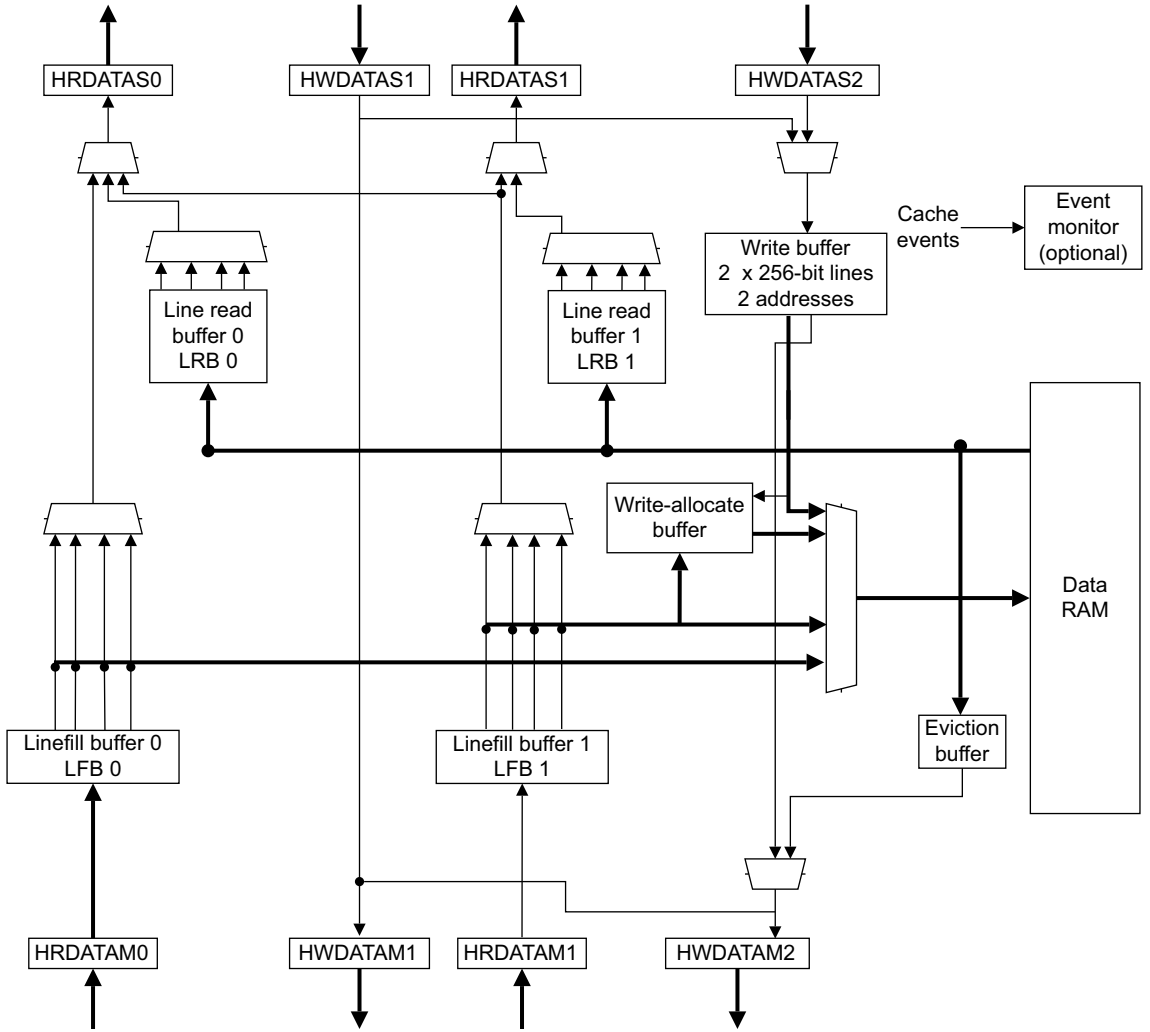


Figure 1-1 Cache controller internal data paths

To enable flexibility, you can configure the cache controller to use one, two, or three ports. The cache controller ports are:

- 64-bit AHB-Lite slave ports
  - One slave, S1
  - Two slaves, S0 and S1
  - Three slaves, S0, S1, and S2.

  The cache controller arbitrates internally among them.
- 64-bit AHB-Lite master ports:
  - One master, M1
  - Two masters, M0 and M1
  - Three masters, M0, M1, and M2.

If you decide not to use S0, you cannot configure M0. S1 always exists in all configurations.

The cache controller master ports are AHB-Lite compatible. ARM Limited can supply an AHB wrapper that converts the AHB-Lite interface to a multi-master AHB interface and that does not introduce any pipeline delays.

1.2.1 Operating frequency

For AHB slaves the clock is **CLK**, the same clock as for internal cache controller logic. For AHB masters the clock is **HCLK**. **CLKEN**s are used as enables for slave ports clocked with **CLK**. **HCLKEN**s are used as enables for master ports clocked with **HCLK**

The CPU AHB ports of the cache controller can operate at the same frequency as the CPU core. This means that the CPU core clock and **CLK** operate in a 1:1 ratio.

AHB arbitration

All AMBA signals are considered valid on a rising clock edge with **HREADY** high and **HCLKEN** high.

Timing diagrams on page 4-20 shows examples of typical cache controller bus transactions.

1.3 Functional description

The L210 Cache Controller is an eight-way unified, physically addressed (indexed), physically tagged cache. The cache controller does not maintain coherency among caches. The cache controller can be locked on a way basis.

The following sections describe the cache controller:

- *Features*
- *Buffers* on page 1-6
- *Error support and MBIST support* on page 1-6
- *Event bus* on page 1-7
- *Configuring the cache controller* on page 1-7.

1.3.1 Features

The features are:

- Size can be 16KB-2MB.
- Fixed line length of 32 bytes, eight words.
- 300 MHz worst case with 0.18 micron technology operating frequency range.
- Physically addressed and physically tagged.
- Lockdown format C supported, with separate way locking mechanisms for data and instructions.
- Eight-way associativity which can be direct mapped, depending on the use of lockdown registers.
- Data RAM is byte-writable.
- Support for these cache modes:
  - Write-Through, read allocate
  - Write-Back, read allocate
  - Write-Back, read and write allocate, for systems using ARMv6 extensions only.
- Write allocate override option to always have allocation on write misses in the cache controller.
- Performs critical word first refilling, with the option of refilling starting with word 0. The same option is used to transform nonbufferable wrap write bursts and noncacheable wrap read bursts into linear accesses.
- Pseudo-random victim selection policy, can be made deterministic with use of lockdown registers.
- You can statically configure the following options:
  - Single master port, master port 1
  - Two master ports, master port 0 and master port 1

  - Three master ports, master port 0, master port 1, and master port 2
- Static option to select synchronous or asynchronous master port interfaces
- Parity generation and error detection synthesis option.

1.3.2 Buffers

The cache controller has the following buffers:

Two linefill buffers, LFB

These buffers capture linefill data from main memory, waiting for a complete line before writing to L2 memory. This makes the cache controller non-blocking for requests from the other slave ports

Two line read buffers, LRB

These buffers hold a line from the cache controller, for subsequent requests that hit on the line.

One eviction buffer, EB

This holds an evicted line from the cache controller, to be written back to main memory.

One write buffer, WB

This holds buffered writes before their draining to main memory, as well as to the cache controller. The write buffer is made of two slots, each with a 256-bit data line and one address per slot. It enables multiple writes to the same line to be merged.

One write-allocate buffer, WA

If the write buffer line is not full, this buffer merges data from the write buffer and missing data, from master port 1 before requesting an allocation to the cache.

1.3.3 Error support and MBIST support

Support is provided for:

- MBIST testing. See Chapter 5 *Design for Test*.
- Optional parity and RAM error detection. See Chapter 6 *Parity and RAM Error Support*.

1.3.4 Event bus

The cache controller has an event bus. The event bus enables licensees to report specific events occurring in the cache controller. It can be used in conjunction with an event monitor so that performance statistics can be extracted. See Appendix C *Event Monitor* for details.

1.3.5 Configuring the cache controller

The cache controller is configured using memory-mapped registers. See Chapter 2 *Programmer's Model* for details.

The cache controller is highly configurable. You do not have to resynthesize the module to reconfigure the cache controller for other applications. This means that you only have to harden the cache controller macrocell once, regardless of the number of cache controller configurations you use.You can use synthesis options to remove S0,S2, M0, and M2. If S0 is not implemented, then M0 must also be removed. S0,S2, M0, and M2, if present, can be disabled using the **MASTNUM** and **SLAVENUM** pins. This means that you can harden a cache controller with all ports and disable unused ones in a specific SoC. S1 always exists in all configurations.

Port configurations

This section describes the different configurations of slave ports and master ports:

- *Slave ports*
- *Master ports* on page 1-8.

Slave ports

There are three requesting ports to the cache controller, the AHB slave ports:

S0 AHB Slave port 0 Only services reads.

S1 AHB Slave port 1

Services both reads and writes, and services swaps. It is the only port that can read the internal cache controller registers. All configurations must support this port

S2 AHB Slave port 2 Only services writes.

All slave ports support exclusive accesses and locked accesses. For more information see *Locked accesses* on page 4-19 and *Exclusive accesses* on page 4-19.

The cache controller can work with three, two, S0 and S1, or one, S1 only, slave ports. You choose the number of slave ports by forcing the correct value on the **SLAVENUM[1:0]** input pins. Clocks of unused slaves are inactive in functional mode (static high-level clock gating).

Master ports

Table 1-2, Table 1-3 on page 1-9, and Table 1-4 on page 1-9 show which transactions each master port is used for in the three possible master port configurations.

The cache controller can work with three, two, M0 and M1, or one, M1 only, master ports. You choose the number of slave ports by forcing the correct value on the **MASTNUM[1:0]** input pins. Clocks of unused slaves are inactive in functional mode (static high-level clock gating).

Table 1-2 shows the transactions on the three master ports in a three-master system. For certain types of transactions the default values for **HMASTERM1** and **HMASTERM2** are configurable in the cache controller RTL, see Appendix A *Signal Descriptions*.

Note

In Appendix A *Signal Descriptions*, the default values for **HMASTERM1** and **HMASTERM2** on certain types of transactions are configured as `0xF`.

Table 1-2 Transactions for a three-master port system

Master port 0	Master port 1	Master port 2
Linefills with linefill buffer 0	Linefills with linefill buffer 1	Buffered stores with eviction buffer
Noncached reads	Noncached reads	Buffered stores with write buffer
-	Swaps, noncached reads andnonbufferable writes	Nonbuffered stores

Table 1-3 shows the transactions on the two master ports in a two-master system.

Table 1-3 Transactions for a two-master port system

Master port 0	Master port 1
Linefills with linefill buffer 0	Linefills with linefill buffer 1
Noncached reads	Noncached reads
-	Swaps, noncached reads andnonbufferable writes
	Buffered stores with eviction buffer
	Buffered stores with write buffer
	Nonbuffered stores

Table 1-4 shows the transactions on the single master port, master port 1, in a single-master port system.

Table 1-4 Transactions for a single-master port system

Master port 1
Linefills with linefill buffer 1 only
All noncached reads
Swaps, noncached reads, andnonbufferable writes
Buffered stores with eviction buffer
Buffered stores with write buffer
Nonbuffered stores

1.4 Supported ARM architectures

The cache controller supports ARM architecture v5 and ARM architecture v6 with ARM1136 AMBA extensions.

ARM processors do not support any form of hardware cache coherency maintenance. All caches have no cache coherency protocol and do no snooping.

1.5 L210 Cache Controller product revisions

This is the Technical Reference Manual for L210 Cache Controller silicon revision r0p5. See *Product revision status* on page xii for details of revision numbering.

There are no differences in functionality between the product revisions of the cache controller. See the engineering errata that accompanies the product deliverables for more information.

Chapter 2
Programmer's Model

This chapter describes the cache controller registers and provides information for programming the macrocell. It contains the following sections:

- *ARM register fields* on page 2-2
- *About the cache controller registers* on page 2-3
- *Register summary* on page 2-4.
- *Replacement strategy* on page 2-19
- *Register 15: Test and Debug* on page 2-20
- *Buffers* on page 2-23
- *Ports configuration* on page 2-25
- *Hazards* on page 2-27
- *External abort support for L3 memory* on page 2-28.

2.1 ARM register fields

All reserved or unused address locations must not be accessed because this can result in unpredictable behavior of the device.

All reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.

All register bits are reset to logic 0 by a system reset unless otherwise stated in the relevant text.

Unless otherwise stated in the relevant text, all registers support read and write accesses. A write updates the contents of the register and a read returns the contents of the register.

All registers defined in this document can only be accessed using word reads and word writes, unless otherwise stated in the relevant text.

2.2 About the cache controller registers

The cache controller is controlled by a set of 32-bit memory-mapped registers that occupy a relocatable 4KB memory space.

The cache controller registers can be written to using slave ports S1 or S2:

- if **HSELRS1** is HIGH and **HSELRS2EN** is LOW, slave port S1 can be written
- if **HSELRS2** is HIGH and **HSELRS2EN** is HIGH, slave port S2 can be written.

If **HSELRS1** is HIGH, S1 can read registers. **HADDRSx[11:2]** is compared to see which register is being accessed.

All registers except some register 15 subregisters accept 32-bits accesses only. The results of 64-bit accesses are Unpredictable.

Register r1, the Auxiliary Control Register, must only be written with a read-modify-write type access when the cache is turned off.

Data line subregisters of register 15 support 64-bit transfer to enable setting or reading registers with `LDM` or `STM` instructions.

Reads to an unmapped register return 0. Table 2-1 shows the cache controller registers.

Table 2-1 Summary register map

Register	Reads	Writes
0	System ID and Cache Type	Ignored
1	Control	Control
2-6	Reserved	Reserved
7	Cache maintenance operations	Cache maintenance operations
8	Reserved	Reserved
9	Cache lockdown	Cache lockdown
10-14	Reserved	Reserved
15	Test and debug	Test and debug

2.3 Register summary

This section describes the following:

- *ID Register* on page 2-7
- *Control Register* on page 2-9
- *Auxiliary Control Register* on page 2-9
- *Cache maintenance operations* on page 2-12
- *Register 9: Cache Lockdown* on page 2-15
- *Uses of Lockdown Format C* on page 2-16.

Table 2-2 shows the cache controller registers, their base addresses, access types, and functions. The base address of the cache controller is not fixed, but is determined by an AHB decoder on the slave buses and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.

Table 2-2 Register map

Register	Name	Base offset	Access type	Reset value	Description
r0	Cache ID	0x000	RO	0x41000040[a]	See *ID Register* on page 2-7
r0	Cache Type	0x004	RO	0x1C100100	See *Cache Type Register* on page 2-8
r1	Control	0x100	-	0x00000000	See *Control Register* on page 2-9
r1	Auxiliary Control	0x104	-	0x00020FFF	See *Auxiliary Control Register* on page 2-9
r7	Cache Sync	0x730	Always *Read As Zero*, RAZ	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Invalidate By Way	0x77C	-	0x00000000	See *Cache maintenance operations* on page 2-12

Table 2-2 Register map (continued)

Register	Name	Base offset	Access type	Reset value	Description
r7	Invalidate Line By PA	0x770	RAZ	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Clean Line by Index Way combination	0x7B8	RAZ	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Clean by Way	0x7BC	-	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Clean Line by PA	0x7B0	RAZ	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Clean and Invalidate Line by Index Way combination	0x7F8	RAZ	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Clean and Invalidate by Way	0x7FC	-	0x00000000	See *Cache maintenance operations* on page 2-12
r7	Clean and Invalidate Line by PA	0x7F0	RAZ	0x00000000	See *Cache maintenance operations* on page 2-12
r9	Lockdown by Way – D Side	0x900	-	0x00000000	See *Register 9: Cache Lockdown* on page 2-15
r9	Lockdown by Way – I Side	0x904	-	0x00000000	See *Register 9: Cache Lockdown* on page 2-15

Table 2-2 Register map (continued)

Register	Name	Base offset	Access type	Reset value	Description
r15	Test Operation	0xF00	RAZ	0x00000000	See *Register 15: Test and Debug* on page 2-20
r15	Line Data (8 x Word)	0xF10	-	0x00000000	See *Register 15: Test and Debug* on page 2-20
r15	Line Tag {Tag, V, D0, D1, RR}	0xF30	-	0x00000000	See *Register 15: Test and Debug* on page 2-20
r15	L2 Debug Control Register	0xF40	-	0x00000000	See *Register 15: Test and Debug* on page 2-20

a. This value is pin dependent, depending on how external CACHEDID pins are tied.

Test and debug is implementation-specific, and must not be used by software.

2.3.1 ID Register

The read-only ID Register returns a 32-bit device ID code. The device id is specified by the value tied on the **CACHEID[5:0]** input.

You can access the ID Register by reading from the cache controller base address + `0x000`. Figure 2-1 shows the format of the ID Register.

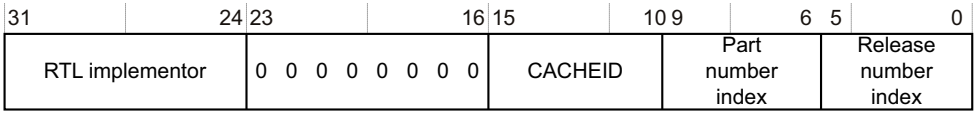


Figure 2-1 ID Register

Table 2-3 shows the encoding of the ID Register.

Table 2-3 ID Register encoding

Register bits	Function	Values
[31:24]	RTL implementor	`0x41`
[23:16]	*Read as zero* (RAZ)	-
[15:10]	Input pins for layout implementor	-
[9:6]	Part number index	`0x1`
[5:0]	RTL release index	See Table 2-4

Table 2-4 shows the release number index values

Table 2-4 Release number index values and releases

Release number index	Release
`0x0F`	r0p5
`0x0B`	r0p4
`0x03`	r0p3
`0x02`	r0p2_01
`0x01`	r0p1
`0x00`	r0p2_02

2.3.2 Cache Type Register

This read-only register returns the 32-bit Cache Type, which makes the cache parameters a product of cache controller cache way size and the cache controller associativity. Figure 2-2 shows the format of the Cache Type Register.

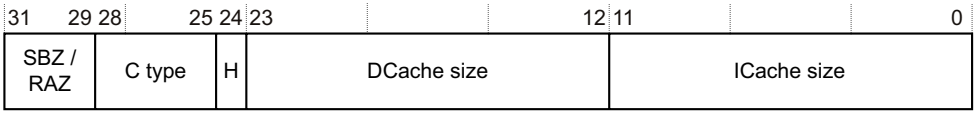


Figure 2-2 Cache Type Register

Table 2-5 shows the encoding of the Cache Type Register.

Table 2-5 Cache Type Register

Bits	Field	Subfield	Comments
[31:29]	*Should be Zero* (SBZ)	-	0b00
[28:25]	Cache type	-	0b1110, Lockdown format C
[24}	H	-	0b0, unified
[23:12]	DCache size	-	-
	[23:20][a]	Way size	Read from Auxiliary Control Register bits [19:17]
	[19:15][b]	Associativity	Read from Auxiliary Control Register bits [16:13]
	[14]	SBZ	-
	[13:12]	Line length	0b00-32 bytes
[11:0]	ICache size	-	-
	[11:8][c]	Way size	Read from Auxiliary Control Register bits [19:17]
	[7:3][d]	Associativity	Read from Auxiliary Control Register bits [16:13]
	[2]	SBZ	-
	[1:0]	Line length	b00-32 bytes

a. The first bit is always 0.
b. The first bit is always 0.
c. The first bit is always 0.
d. The first bit is always 0.

Sizes and associativity on page 4-2 gives more information on cache size and associativity.

2.3.3 Control Register

The Control Register, register 1, must be accessed using a read-modify-write sequence. Figure 2-3 shows the format of the Control Register.

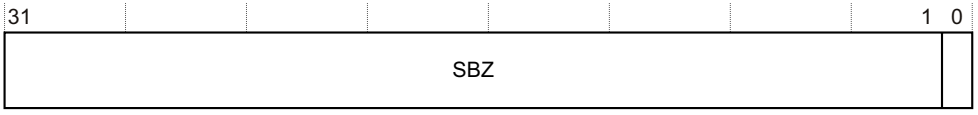


Figure 2-3 Control Register

Table 2-6 shows the encodings of the Control Register.

Table 2-6 Control Register

Bit	Field	Description
[31:1]	Reserved	SBZ
[0]	Unified cache enable	0 = Cache in bypass mode, default 1 = Cache is enabled.

Note

Any change to the cache enable bit is seen by slave ports only at the start of a new transaction. If the bit is changed while a transaction is on going on a slave port, that transaction completes as if the enable bit did not change, so an ongoing linefill, and possible subsequent line eviction, completes even if the cache is turned off.

2.3.4 Auxiliary Control Register

Figure 2-4 on page 2-10 shows the format of the Auxiliary Control Register.

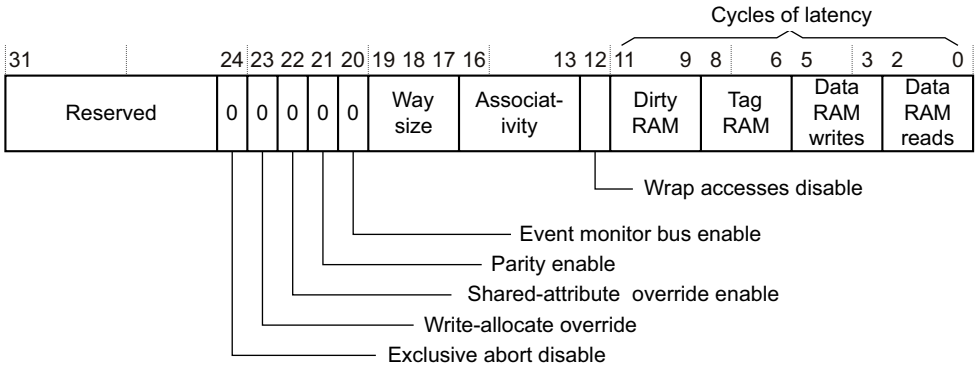


Figure 2-4 Auxiliary Control Register format

Table 2-7 shows the encodings for the Auxiliary Control Register.

Table 2-7 Auxiliary Control Register

Bits	Field	Description
[31:25]	Reserved	-
[24]	Exclusive abort disable	0 = The cache controller sends an ERROR response back to exclusive access in a cacheable, shared memory region with shared override bit set, default 1 = abort generation for exclusive access disabled. Treated as cacheable non-shared accesses.
[23]	Write allocate override	0 = Use of **HPROT** attributes, default 1 = Override **HPROT** attributes. All Write-Through and write-back accesses are read-write-allocate.
[22]	Shared attribute override enable	0 = Shared accesses treated as noncacheable, default 1 = Shared attribute internally ignored but still forwarded to L3 memory.
[21]	Parity enable	0 = Disabled, default 1 = Enabled.
[20]	Event bus enable	0 = Disabled, default 1 = Enabled.

Table 2-7 Auxiliary Control Register (continued)

Bits	Field	Description
[19:17]	Way size	0b000 Reserved, and internally mapped to 16KB 0b001 16KB, default 0b010 32KB 0b011 64KB 0b100 128KB 0b101 256KB 0b110-0b111 Reserved, and internally mapped to 256KB.
[16:13]	Associativity	0b0000 Cache absent, default 0b0001 direct-mapped cache 0b0010 2-way cache 0b0011 3-way cache 0b0100 4-way cache 0b0101 5-way cache 0b0110 6-way cache 0b0111 7-way cache 0b1000 8-way cache 0b1001-0b1111 Reserved, and internally mapped to 8-way associativity.
[12]	Wrap, accesses disable	0 = Master ports can perform wrap accesses, default. 1 = Wrap accesses requested on slave ports are converted to linear accesses on master ports.
[11:9]	Latency for dirty RAM	0b000 1 cycle of latency, no additional latency 0b001 2 cycles of latency 0b010 3 cycles of latency 0b011 4 cycles of latency 0b100 5 cycles of latency 0b101 6 cycles of latency 0b110 7 cycles of latency 0b111 8 cycles of latency, default. The output signals **DIRTYLAT[2:0]**, **TAGLAT[2:0]**, **WDATALAT[2:0]**, and **RDATALAT[2:0]** reflect the values set in the Auxiliary Control Register in the respective fields. See *Miscellaneous signals* on page A-16.
[8:6]	Latency for tag RAMs	
[5:3]	Latency for data RAM writes	
[2:0]	Latency for data RAM reads	

2.3.5 Cache maintenance operations

The cache maintenance operations groups all Register 7 subregister functions. Table 2-8 shows cache maintenance operations.

Table 2-8 Cache maintenance operations

Operation	Description	Bit assignment format
Cache Sync	Drain write buffer (WB) and eviction buffer (EB) to L3. Drain write-allocate buffer (WA) to the data RAM. The Cache Sync operation is considered to be complete when the write buffer (WB), eviction buffer (EB) and write-allocate buffer (WA) are empty, regardless of on-going linefills that could generate a new eviction.	N/A
Invalidate by Way	Invalidate all data in specified ways including dirty data. Completes as a background task. Invalidate All operation is equivalent to Invalidate by way while selecting all cache ways. Completes as a background task with the selected way or ways locked preventing allocation.	See Figure 2-7 on page 2-15
Invalidate Line by PA	Specific line is marked as not valid.	See Figure 2-5 on page 2-14
Clean Line by Index Way combination	Write the specific line within the specified way to main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.	See Figure 2-6 on page 2-14
Clean by Way	Writes each line of the specified ways to main memory if the line is marked as valid and dirty. The lines are marked as not dirty. The valid bits are unchanged. Completes as a background task with the selected way or ways locked preventing allocation.	See Figure 2-7 on page 2-15
Clean Line by PA	Write the specific line to main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.	See Figure 2-5 on page 2-14

Table 2-8 Cache maintenance operations (continued)

Operation	Description	Bit assignment format
Clean and Invalidate Line by Index Way combination	Write the specific line within the specified way to main memory if the line is marked as valid and dirty. The line is marked as not valid.	See Figure 2-6 on page 2-14
Clean and Invalidate Line by PA	Write the specific line to main memory if the line is marked as valid and dirty. The line is marked as not valid.	See Figure 2-5 on page 2-14
Clean and Invalidate by Way	Writes each line of the specified ways to main memory if the line is marked as valid and dirty. The lines are marked as not valid. Completes as a background task with the selected ways locked preventing allocation.	See Figure 2-7 on page 2-15

A Cache Sync operation is always performed automatically when an atomic or background cache maintenance operation is requested.

Note

Cache maintenance operations are always performed whatever the state of the cache enable bit, bit 0 of the Control Register, regardless of whether the cache is on or off.

Atomic operations

The following operations stall the requesting slave ports until they complete:

- Clean Line
- Invalidate Line
- Clean and Invalidate Line
- Cache Sync.

Background operations

The following operations run as background tasks. The subregister dedicated to the maintenance operation must be polled to see when a background cache operation has completed:

- Invalidate by Way
- Clean by Way
- Clean and Invalidate by Way.

Requesting slave ports are not stalled during the operation. Table 2-8 on page 2-12 shows what might happen. A background way maintenance operation does not prevent the cache controller from accessing that specific way. For example, accessing data in a line that was in the cache before the start of an Invalidate by Way operation, can cause a cache hit or not, depending on the progress of the background operation. In all cases the cache controller deals with the potential hazards and maintains data coherency.

If an atomic or a background task is requested before a previously requested task has completed, the second request is ignored.

During background operation the targeted way is considered locked. This means that no allocation occurs to that way on read or write misses.

Line-based and way-based cache maintenance operations

The cache maintenance operations are executed by writing to the cache operations register r7.

For cache maintenance operations on lines, the line can be accessed using:

- Physical Address
- Index Way combination.

For line-based operations, the PA or Index Way combination is given on **HWDATASx[31:0]**. The physical address format is shown in Figure 2-5.

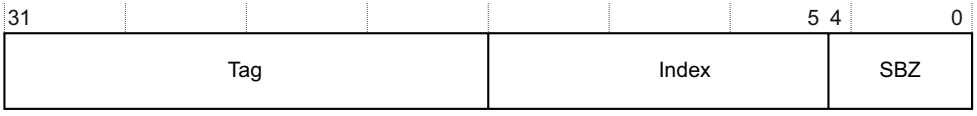


Figure 2-5 Physical address format

Note

Because the cache is configurable, the Index field can be one of several bit sizes. This is why it is undefined.

Figure 2-6 shows The Index Way combination format.

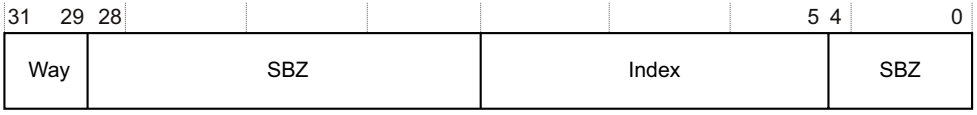


Figure 2-6 Index Way combination format

Cache maintenance operations are atomic operations, and writing to the register starts the operation, on the line specified by either {Tag, Index} or {Way, Index}. Tag and Index fields sizes depend on cache way size. See *RAM configuration versus associativity and way size* on page 3-3. The requesting slave port, S1 or S2, remains blocked until the operation completes. Those registers are always read as 0.

For way-based operations, the way is given on **HWDATASx[31:0]**, using the Format C lockdown. Multiple ways can be selected at the same time, by setting the way bits to one. Figure 2-7 shows the format for Format C lockdown.

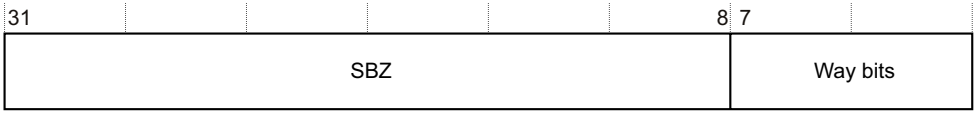


Figure 2-7 Lockdown format C

Way-based operations are background operations, so writing to the register starts the operation, on the ways set to one in bits [7:0]. The Way bits are reset to 0 as the respective ways are cleaned and/or invalidated, so the Register 7 subregister must be polled to see when the operation has completed.

Table 2-9 shows the operations that can be performed on the way bits:

Table 2-9 Writing or reading register to way bits

Operation	Description
Writing 0 to bit[n]	Do not perform operation on way n
Writing 1 to bit[n]	Perform operation on way n
Reading 0 in bit[n]	No operation or a maintenance operation completed on way n
Reading 1 in bit[n]	A maintenance operation is ongoing or not completed on way n

2.3.6 Register 9: Cache Lockdown

Cache lockdown is controlled by the Cache Lockdown Register, r9. It is implemented as two lockdown subregisters, one for data and one for instructions.

If a cache lookup misses and a cache linefill is required, there are eight possible locations where the new line can be placed. Lockdown format C restricts the cache replacement algorithm to only use a subset of the eight possible locations. The choice of an eight-way associativity for the cache controller increases the hit rate and increases the effectiveness of Lockdown Format C. With Lockdown Format C, a block of the cache controller can be used as a form of frame buffer.

If all ways are locked, a linefill is performed on a cache miss, reading eight words from external memory, but the cache is not updated with the linefill data. In the same way, Write-Through write allocate and Write-Back write allocate accesses are treated as normal Write-Through and Write-Back accesses respectively.

2.3.7 Uses of Lockdown Format C

Lockdown format C, used in the ARM926EJ-S, ARM1026, and ARM1136 caches, provides a method to restrict the replacement algorithm used on cache linefills and to only use selected cache ways within a set. Using this method, code can be fetched or loaded into the cache controller and protected from being evicted.

- Use lockdown format C to restrict the available ways for cache filling to n-ways, where n is less than the total number of ways, for example by only enabling filling to way 0.
- Use lockdown format C to fetch code into the cache:
  - — Executing the routine for the first time. The I Lockdown Register is used.
  - — Loading the code into the cache, using a noncacheable load routine. The D Lockdown Register used as the code is loaded as data.
- Use lockdown format C to load data into the cache by:
  - — Loading data into the cache for the first time. The D Lockdown Register is used in both cases.
  - — Loading data into the cache, using a noncacheable load routine.
- Write to the lockdown register to prevent filling to way 0, but enable filling to ways 1-7. The code and data is now protected in the cache and is not evicted on linefill.

Note

A noncacheable routine is a software loop noncacheable at L2 in order to prevent cache pollution. It is executed by the CPU, and reads at least one word out of 8, L2 cache line size, in the data pool or code pool that is intended to be put in the L2 cache.

Preventing or reducing cache pollution

There can be benefits in having a critical piece of software or data being cached at the cache controller with the insurance that it cannot be polluted or evicted, due to a new allocation. Using lockdown format C provides a simple method to load data into the

cache controller using the linefill mechanism, then locking the cache memory to prevent eviction, and finally using the cache controller cache maintenance operations to efficiently clean the data to the main memory system.

To do that:

1. Use lockdown format C (I and/or D lockdown) to restrict the permitted ways for cache filling to n-ways, where n is less than the total number of ways. For example only permit filling to way 0.
2. Cache code or data into the cache
   - Fetch code into the cache by executing the routine for the first time.
   - Load data into the cache by:
     — loading data for the first time
     — cache the data in the cache controller by executing the read loop being cached at L1 only.
3. Write to the lockdown register, I and/or D lockdown, to prevent allocation to way 0, but enable allocation to ways 1-7. The code or data is now protected in the cache and cannot be evicted on a linefill.

Using Lockdown Format C with the cache controller for processing frame buffers

There might be benefits in processing large frame buffers in the cache controller, and making them appear as if there is a large amount of restricted physically addressed space available in fast memory. Because the cache controller is eight-way set associative, using lockdown format C provides a simple method to:

1. Load data into the cache controller using the linefill mechanism
2. Lock the cache memory to prevent eviction
3. Use the cache controller cache maintenance operations to efficiently clean the data to the main memory system.

Consider the example of requiring a 1MB frame buffer in the 2MB 8-way set associative cache controller, in a system using ARM1136 AMBA extensions with ARMv6 architecture.

1. Set the address page attributes so the L1 page is noncacheable and the L2 page is cacheable.
2. Set the cache controller lockdown to only fill to ways 0-3 (=1MB).
3. The 1MB is now contiguously mapped over four ways of 256KB each.

4. Load data into the cache controller by executing a load routine in an ARM1136 core, where a series of LDRs are issued, one cache line apart from one another. The L1 memory attribute is noncacheable, so the L1 cache is not polluted. The cache controller memory attribute is cacheable, so the cache controller performs linefills, filling into ways 0-3. The linefill buffer in the cache controller provides efficient filling.

5. When finished, set the cache controller lockdown to lock ways 0-3, and only enable filling to ways 4-7 (=1MB).

The cache controller now contains a 1MB frame buffer, and 1MB of four-way associative cache. Lookups and reads or writes can occur to the entire 2MB, but the frame buffer is prevented from being evicted. Cache maintenance operations can be used to efficiently clean to main memory. When the frame buffer is no longer required, ways 0-3 can be unlocked, and are naturally overwritten. Lockdown format C has a pattern-matching field, so any of the eight ways can be locked. There is no requirement to start at 0 and work up. If all ways are marked as being locked, then nothing is allocated.

2.4 Replacement strategy

The cache controller uses a pseudo-random replacement strategy. When used in combination with the lockdown registers, a deterministic replacement strategy can be achieved.

The pseudo-random replacement strategy fills empty, unlocked ways first. If a line is completely full, the victim is chosen as the next unlocked way.

If a deterministic replacement strategy is required, the lockdown registers are used to prevent ways from being allocated. For example, if the L2 size is 256KB, each way is 32KB, and the user wants a piece of code to reside in two ways, 64KB space, with a deterministic replacement strategy, ways one-7 must be locked before the code is filled into the cache controller. The first 32KB of code is allocated like this into way 0 only. Then, way 0 must be locked and way one unlocked so that the second half of the code is allocated in way one.

There are two lockdown registers, one for data and one for instructions, so the user can also separate data and instructions into separate ways of the cache controller, if required.

2.5 Register 15: Test and Debug

The Test Operation Register enables the contents of the cache controller to be read and written.

For cache line read, test operation is written with the nRW bit cleared so that the content of the line designated by the Index Way combination fields is put in the line data registers and its attributes put in the Line Tag Register. All information needed about the cache line can then be retrieved by reading Line Data and Line Tag Registers.

For cache line writes, all line data and attributes must be written first in both the Line Data Registers and Line Tag Registers. At this time, by writing to the Test Operation Register, the cache line designated by the Index Way combination fields is updated with register contents.

The Test Operation Register is stored in Index Way combination format as shown in Figure 2-8. Index field size depends on cache way size.

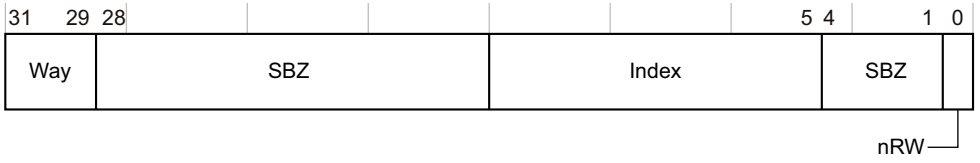


Figure 2-8 Test operation format

Figure 2-9 shows the order in which words are stored in a cache line.

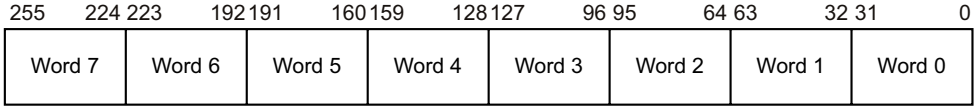


Figure 2-9 Word order in cache lines

Figure 2-10 shows the Line Tag Register.

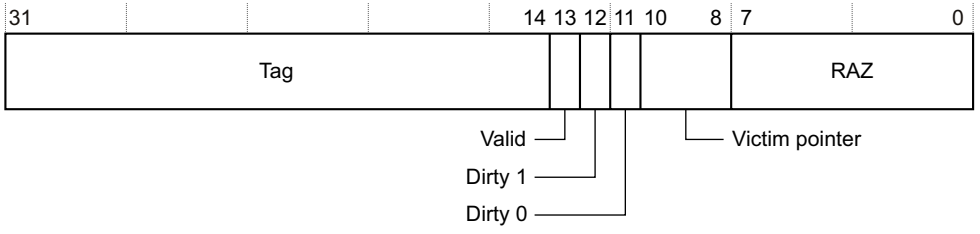


Figure 2-10 Line Tag Register format

Table 2-10 shows the Line Tag Register mapping.

Table 2-10 Line Tag Register mapping

Bits	Description
[31:14]	Tag An invalid line always has its tag set to zero until it has been written through a test operation
[13]	Valid
[12]	Dirty 1 Defines state of the last four words in the cache line, words 4-8
[11]	Dirty 0 Defines state of the first four words in the cache line, words 0-3
[10:8]	Victim pointer Defines last allocated way b111 represents way 7 b000 represents way 0
[7:0]	RAZ

Address BASE + `0xF10` stands for Word 0 in the line, up to Base + `0xF2C` which stands for Word 8.

2.5.1 Debug Control Register

The Debug Control Register forces specific cache behavior required for debug. Table 2-11 shows the Debug Control Register bit assignments.

Table 2-11 Debug Control Register

Bit	Field	Description
[31:2]	Reserved	SBZ/RAZ
[1]	DWB Disable Write-Back, force WT	0 = Enable Write-Back behavior, default. 1 = Force Write-Through and read-allocate only behavior
[0]	DCL Disable cache linefill	0 = Enable cache linefills, default 1 = Disable cache linefills

Forcing Write-Through read-allocate only behavior

Setting the DWB bit to 1 forces the cache controller to treat all cacheable accesses as though they are in a Write-Through read-allocate only region of memory. Setting the DWB bit overrides access attributes. If the cache contains dirty cache lines, these remain dirty while the DWB bit is set, unless they are written back because of a Write-Back eviction after a linefill, or because of an explicit clean operation. Lines that are clean are not marked as dirty if they are updated while the DWB is set. This functionality enables a debugger to download code or data to external memory, without the requirement to clean part or the entire cache to ensure that the code or data being downloaded has been written to external memory.

If the DWB is set, and a write is made to a cache line that is dirty, then both the cache line and the external memory are updated with the write data. Other entries in the cache line still have to be written back to main memory to achieve coherency.

Disabling cache linefills

Setting the DCL bit prevents the cache from updating when performing a linefill on a miss. When set, a linefill is performed on a cache miss, reading eight words from external memory, but the cache is not updated with the linefill data. This mode of operation is required for debug so that the memory image, as seen by the CPU core, can be examined in a noninvasive manner. Cache hits read data words from the cache, and cache misses from a cacheable region read words directly from memory.

Setting the DCL bit overrides the write-allocate attributes. Write-Through write allocate and Write-Back write allocate accesses are treated as normal Write-Through and Write-Back accesses respectively.

2.6 Buffers

This section describes the following buffers:

- *Write buffer*
- *Write-allocate buffer*
- *Eviction buffer* on page 2-24.

2.6.1 Write buffer

The write buffer has two slots that each contain one 256-bit data line and its associated address. The write buffer has merging capabilities so that successive writes to the same line address are merged in the same buffer slot. Two buffered write accesses to the same address cause the first one to be overridden if the write buffer has not been drained in between the writes.

Merging capability means that lines are not treated as soon as they contain data. Write buffer draining policy is:

- the write buffer is drained at each noncacheable read occurrence
- the write buffer is drained at each nonbufferable write occurrence
- if the two slots of the write buffer contain data, the least recently accessed is drained
- if a hazard is detected with one write buffer slot, it is drained to resolve the hazard.

Each slot contains a byte-valid field that allows the control logic to know the data line fill level. If a drained slot is drained while its data line is not full, the write buffer must request correct transactions to the master port in the form of bursts or linear access requests.

For write allocate transactions, the write buffer transfers information for one of its slots to a write-allocate buffer described in *Write-allocate buffer*.

2.6.2 Write-allocate buffer

Allocation to the cache in case of write-allocate transaction is not performed directly by the write buffer. In cases of a write miss, when a write buffer slot is drained the data, address, and byte-valid information is sent to the write allocate buffer. Write allocate behavior is deduced from the L2 memory attributes of a transaction and from the write allocate override bit in the Auxiliary Control Register. See *Auxiliary Control Register* on page 2-9.

Based on byte-valid information, the write allocate buffer requests, through the M1 port to L3, the data required to fill its data line if not full:

- if the line is full, no request is made.

- if one to eight sequential bytes within the same doubleword boundary are missing, one SINGLE 64-bit transaction is performed on L3
- in all other cases, a full linefill request is performed.

Data from the write buffer and M1 master port are then merged in the write-allocate buffer that can then request an allocation to the cache.

If M1 receives an ERROR response during a read transfer for the write-allocate buffer, the allocation from the write-allocate buffer to the cache is not performed.

2.6.3 Eviction buffer

The eviction buffer is incorporated for holding Write-Back data for cache controller line evictions or cleaning of dirty cache lines. It holds two halves of a cache controller line (32 bytes in total) and two address entries.

2.7 Ports configuration

To simplify AMBA ports design, some of their characteristics have to be configured using the following pins:

- **MASTNUM**
- **SLAVENUM**
- **SIZES0**
- **SIZES1**
- **SIZES2**
- **SIZEM0**
- **SIZEM1**
- **SIZEM2**.

Those characteristics are:

Number of master ports, if present after synthesis

The cache controller can work with three, two, M0 and M1, or one, M1 only, master ports. The choice is made by forcing the correct value on **MASTNUM[1:0]** input pins.

MASTNUM[1:0] is a static input. If its value is changed while the cache controller is in use, even if it is disabled, the result is Unpredictable.

Number of slave ports, if present after synthesis

The cache controller can work with three, two, S0 and S1, or one, S1 only, slave ports. The choice is made by forcing the correct value on the **SLAVENUM[1:0]** input pins.

SLAVENUM[1:0] is a static input. If its value is changed while the cache controller is in use, even if it is disabled, the result is Unpredictable.

Data bus sizes

32-bit wide or 64-bit wide data buses can be configured on a per-port basis. Six input pins, **SIZES0**, **SIZES1**, **SIZES2**, **SIZEM0**, **SIZEM1**, and **SIZEM2** can be forced to configure all three slave and three master ports independently. These are static inputs. If their values are changed while the cache controller is in use, even if it is disabled, the result is Unpredictable.

Endianness The cache controller must be aware of the endianness of the system in which it sits. The cache controller **BIGEND** input pin must be tied to the correct value.

BIGEND is a static input. If its value is changed while the cache controller is in use, even if it is disabled, the result is Unpredictable.

2.8 Hazards

When dealing with L1 hazards, the following assumption is made:

- *Read after Write* (RAW) hazards are handled at the CPU level so that writes arrive at the L2 interface at least one cycle before the read.

2.8.1 L2 hazards handled internally

The following known L2 hazards are all handled internally by the cache controller:

- Read when in WB.
- Read when in EB.
- Read from S0 or S1, when in LFB1 or LFB0. This scenario also describes two reads, both arriving at the same time on different ports, to the same address.
- Read when in WA.
- Write when in LR0 or LR1.
- Write when in LFB0 or LFB1.
- Write when in EB.

2.9 External abort support for L3 memory

The cache controller gives limited support for external aborts, because of restrictions of the AHB protocol. There are methods to acknowledge all external aborts created by a slave device:

- The abort-generating slave must keep a copy of the aborting address. This means the master, after receiving the external abort, whether precise or imprecise, can deduce the aborting address by reading the appropriate slave register.
- The cache controller is a slave to an ARM1136 core, but a master to the main memory system. In effect this means that the cache controller is just a method to communicate between the master core and the slave main memory.
- The AHB protocol does not provide a method for passing back an ERROR response that is not combined with its original transaction.

The following support is supplied by the cache controller, assuming the slave main memory holds a copy of the aborting address.

If an ERROR response is received by the cache controller on a master port **HRESPMx**:

- If this request was a noncacheable read, or nonbufferable write, using ARM v5 nomenclature, then the associated slave port has been waited until the response was received from the main memory. The ERROR response is passed through to L1 on the appropriate word.
- If the request was a cacheable read which missed at L2, then the ERROR response is passed through to L1 on the appropriate word, and the line is not marked as valid in the L2.
- If the request was a bufferable write, then the slave port completes its transaction before the ERROR response is received from main memory. The AHB protocol does not permit this response to be passed back to L1. In this case, the cache controller produces a pulse on the Event bus. See Appendix C *Event Monitor* for details. If there is an event monitor, it can then produce an interrupt to the L1 core.

In this way, all L3 external aborts can be detected at level 1.

Note

If a noncacheable known-length burst is early terminated on a slave port, the behavior of the master port handling the transaction is unpredictable.

Chapter 3
RAM Interfaces

This chapter describes the organization of the cache controller RAM interfaces and RAM sizes. It contains the following sections:

- *About RAM interfaces* on page 3-2
- *RAM configuration versus associativity and way size* on page 3-3
- *Data RAM* on page 3-7
- *Tag RAM* on page 3-9
- *Dirty RAM* on page 3-11.

3.1 About RAM interfaces

Your choice of way size and associativity determines the size and organization of the RAMs and bus usage.

3.2 RAM configuration versus associativity and way size

RAM configuration depends on cache way size and associativity. This section describes the associativity, connections and way size for:

- *Data RAM*
- *Tag RAM* on page 3-4
- *Dirty RAM* on page 3-5.

3.2.1 Data RAM

Figure 3-1 shows the data RAM bus address format.

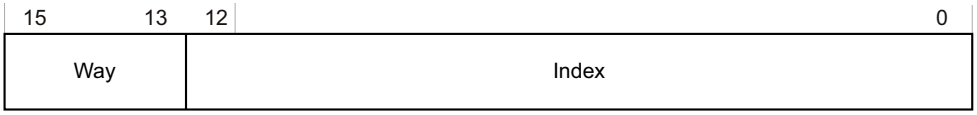


Figure 3-1 Data RAM bus address format

Bits [15:13] connections depend on cache associativity. Table 3-1 shows this.

Table 3-1 Cache associativity and bits[15:13] of data RAM bus address

Associativity	Connections
One or two way	Bits [15:14] are left unconnected. Bit 13 is the *Most Significant Bit* (MSB) of the RAM address bus.
Three or four way	Bit 15 is left unconnected. Bits [14:13] are MSBs of RAM address bus.
Five to eight way	Bits [15:13] are MSBs of RAM address bus.

Bits [12:0] connections depend on way size. Table 3-2 shows this.

Table 3-2 Way size and bits[12:0] of data RAM bus address

Way size	Connections
16KB	Bits [12:9] are left unconnected. Bits [8:0] are the *Least Significant Bits* (LSBs) of RAM address bus.
32KB	Bits [12:10] are left unconnected. Bits [9:0] are LSBs of RAM address bus.
64KB	Bits [12:11] are left unconnected. Bits [10:0] are LSBs of RAM address bus.
128KB	Bit 12 is left unconnected. Bits [11:0] are LSBs of RAM address bus.
256KB	Bits [12:0] are LSBs of RAM address bus.

All bits of the data bus are always connected.

3.2.2 Tag RAM

TAGADDR, TAGRDn, and TAGWDn usage depends on way size. Table 3-3 shows this.

Table 3-3 Tag RAM and way size

Way size	**TAGADDR and TAGRDn, TAGWDn usage**
16KB	Bits [12:9] of **TAGADDR** are left unconnected. Bits [8:0] are the RAM address bus. All bits of **TAGRDn** and **TAGWDn** are used for data buses.
32KB	Bits [12:10] of **TAGADDR** left unconnected. Bits [9:0] are RAM address bus. Bits [18:1] of **TAGRDn** and **TAGWDn** are used for data buses. Bit 0 of **TAGRDn** must be tied LOW.

Table 3-3 Tag RAM and way size (continued)

Way size	TAGADDR and TAGRDn, TAGWDn usage
64KB	Bits [12:11] of **TAGADDR** are left unconnected. Bits [10:0] are RAM address bus. Bits [18:2] of **TAGRDn** and **TAGWDn** are used for data buses. Bit [1:0] of **TAGRDn** must be tied LOW.
128KB	Bit 12 of **TAGADDR** is left unconnected. Bits [11:0] are RAM address bus. Bits [18:3] of **TAGRDn** and **TAGWDn** are used for data buses. Bit [2:0] of **TAGRDn** must be tied LOW.
256KB	Bits [12:0] are RAM address bus. Bits [18:4] of **TAGRDn** and **TAGWDn** are used for data buses. Bit [3:0] of **TAGRDn** must be tied low.

3.2.3 Dirty RAM

Depending on way size, some bits of **TAGADDR** bus are not used for the dirty RAM address. Table 3-4 shows this.

Table 3-4 Way size and TAGADDR values

Way size	Connections
16KB	Bits [12:9] are left unconnected. Bits [8:0] are the RAM address bus.
32KB	Bits [12:10] are left unconnected. Bits [9:0] are the RAM address bus.
64KB	Bits [12:11] are left unconnected. Bits [10:0] are the RAM address bus.
128KB	Bit 12 is left unconnected. Bits [11:0] are the RAM address bus.
256KB	Bits [12:0] are the RAM address bus.

Depending on cache associativity, some bits of **DIRTYRD** bus must be tied LOW and some bits of **DIRTYWD** are left unconnected. Table 3-5 shows this.

Table 3-5 Associativity and DIRTYRD and DIRTYWD values

Associativity	DIRTYRD and DIRTYWD values
One-way	Bits [1:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:2]** must be tied LOW.
Two-way	Bits [3:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:4]** must be tied LOW.
Three-way	Bits [5:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:6]** must be tied LOW.
Four-way	Bits [7:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:8]** must be tied LOW.
Five-way	Bits [9:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:10]** must be tied LOW.
Six-way	Bits [11:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:12]** must be tied LOW.
Seven-way	Bits [13:0] of **DIRTYRD** and **DIRTYWD** used for data buses. Bits **DIRTYRD[15:14]** must be tied LOW.
Eight-way	All bits of **DIRTYRD** and **DIRTYWD** are used for data buses.

3.3 Data RAM

The data RAM is organized as eight ways of 256-bit wide contiguous memories. It supports the following accesses:

- 8-word data reads
- 8-word data writes for linefills.
- 8-bit, 16-bit, 32-bit, and 64-bit data writes for non-sequential or sequential writes with byte lane controls

Figure 3-2 shows the data RAM signals.

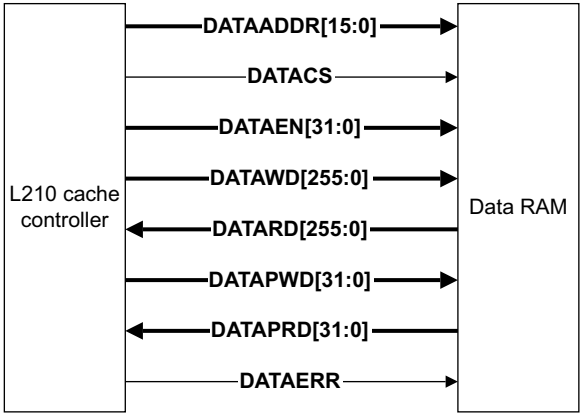


Figure 3-2 Data RAM signals

In all the following diagrams N is used to show the sizes of the RAMs. Table 3-6 shows how N is calculated.

Table 3-6 Values for N in RAM size diagrams

Size	Value for N
128KB	512
256KB	1 024
512KB	2 048
IMB	4 096
2MB	8 192

Figure 3-3 shows the data RAM ways.

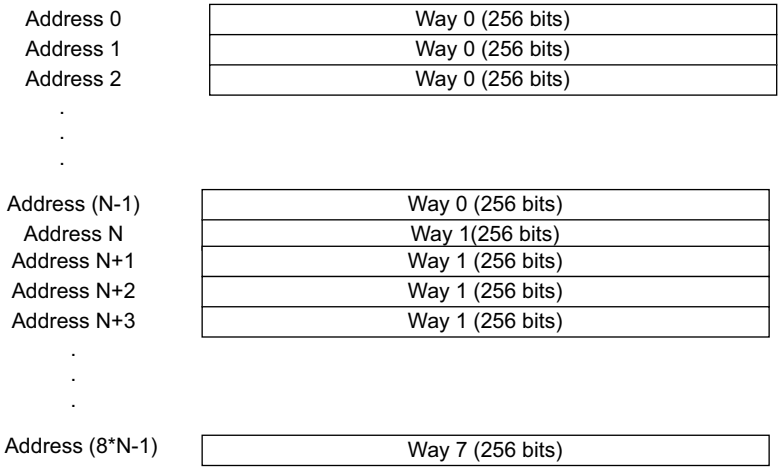


Figure 3-3 Data RAM ways

3.4 Tag RAM

Each way of the cache controller has one tag RAM. The tag RAM is organized as 19-bit wide, maximum, down to 15-bit wide memory:

- 18 bits maximum for the address tag
- one bit for valid information.

The tag RAM address bus is also the address bus for the dirty RAM. The tag RAMs support the following accesses:

- 19-bit tag reads for tag lookup
- 19-bit tag writes for linefills.

Figure 3-4 shows the tag RAM signals,

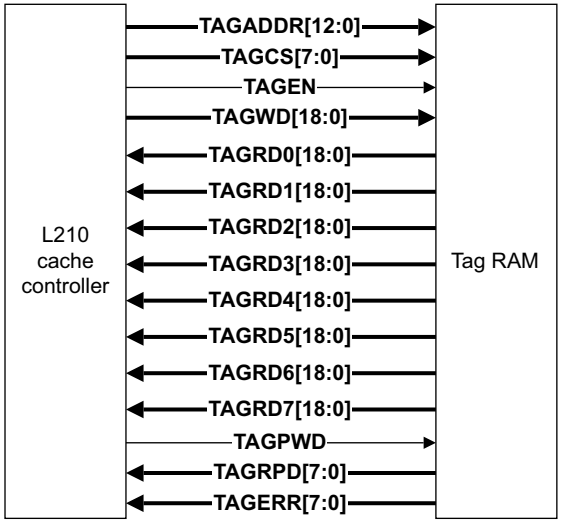


Figure 3-4 Tag RAM signals

Figure 3-5 on page 3-10 shows how the tag RAM is organized. There is one tag RAM for each way.

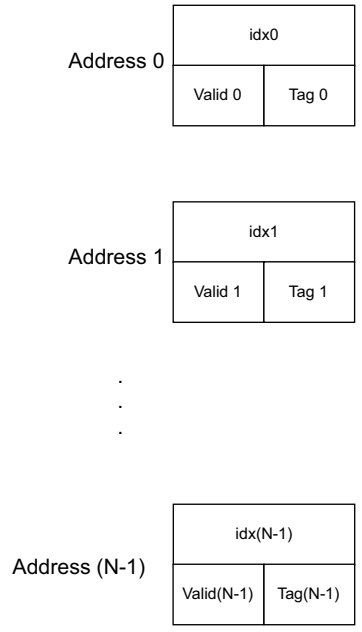


Figure 3-5 Tag RAM organization

3.5 Dirty RAM

The dirty RAM is organized as a 16-bit wide memory with two bits per 8-word cache line as shown in *Dirty RAM memory organization* on page 3-12. The dirty RAM address is the same as the tag RAMs address bus. It supports the following accesses:

- 16-bit dirty reads for write-back on a linefill
- 16-bit dirty reads for cache maintenance operations
- 1-bit or 2-bit dirty writes for writes.

Figure 3-6 shows the dirty RAM signals.

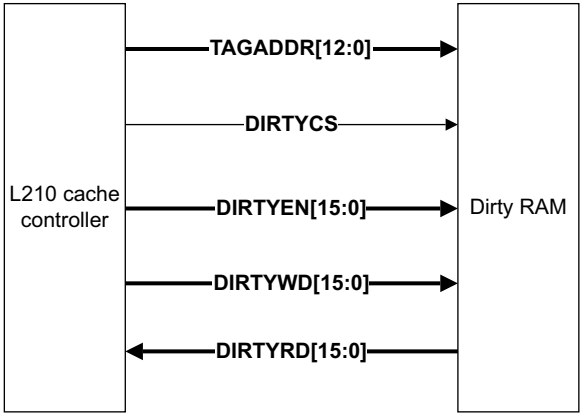


Figure 3-6 Dirty RAM signals

Figure 3-7 on page 3-12 shows how the dirty RAM memory is organized.

	Way 7 words 7-4	Way 7 words 3-0	Way 6 words 7-4	Way 6 words 3-0	Way 5 words 7-4	Way 5 words 3-0	Way 4 words 7-4	Way 4 words 3-0	Way 3 words 7-4	Way 3 words 3-0	Way 2 words 7-4	Way 2 words 3-0	Way 1 words 7-4	Way 1 words 3-0	Way 0 words 7-4	Way 0 words 3-0
Address 0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0	idx0
	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty

	Way 7 words 7-4	Way 7 words 3-0	Way 6 words 7-4	Way 6 words 3-0	Way 5 words 7-4	Way 5 words 3-0	Way 4 words 7-4	Way 4 words 3-0	Way 3 words 7-4	Way 3 words 3-0	Way 2 words 7-4	Way 2 words 3-0	Way 1 words 7-4	Way 1 words 3-0	Way 0 words 7-4	Way 0 words 3-0
Address 1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1	idx1
	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty

.
.
.

	Way 7 words 7-4	Way 7 words 3-0	Way 6 words 7-4	Way 6 words 3-0	Way 5 words 7-4	Way 5 words 3-0	Way 4 words 7-4	Way 4 words 3-0	Way 3 words 7-4	Way 3 words 3-0	Way 2 words 7-4	Way 2 words 3-0	Way 1 words 7-4	Way 1 words 3-0	Way 0 words 7-4	Way 0 words 3-0
Address (N-1)	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1	idxn-1
	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty	Dirty

Figure 3-7 Dirty RAM memory organization

Chapter 4
Using the Cache Controller

This chapter describes how to perform a range of cache controller operations, together with information that you require to use it correctly. It contains the following sections:

- *Configuring the cache controller* on page 4-2
- *Clocking* on page 4-6
- *Idle* on page 4-7
- *Disabled* on page 4-8
- *Set-up sequence* on page 4-9
- *Behavior for ARMv5 memory systems* on page 4-10
- *Behavior for ARMv6 memory systems* on page 4-14
- *Timing diagrams* on page 4-20.

4.1 Configuring the cache controller

Configuring the cache controller is described under:

- *Sizes and associativity*
- *Compiled RAM latency*
- *Locked and exclusive access* on page 4-18
- *Cache Sync* on page 4-3
- *Asynchronous interface* on page 4-4.

4.1.1 Sizes and associativity

The total size of the cache controller can be from 16KB-2MB, determined by the choice of cache associativity and way size. Table 4-1 shows the possible cache sizes and number of locked ways. For example, a way size of 64KB with four-way associativity gives a total cache size of 256KB and a way size of 32KB with eight-way associativity also gives a total cache size of 256KB. An eight-way associativity is likely to give a better hit rate for most applications.

Table 4-1 Total cache size compared to way size and associativity

Way size	Associativity							
	Direct mapped (1-way)	2-way	3-way	4-way	5-way	6-way	7-way	8-way
16KB	16KB	32KB	48KB	48KB	48KB	96KB	112KB	128KB
32KB	32KB	64KB	96KB	96KB	96KB	192KB	224KB	256KB
64KB	64KB	128KB	192KB	192KB	192KB	384KB	448KB	512KB
128KB	128KB	256KB	384KB	384KB	384KB	768KB	896KB	1MB
256KB	256KB	512KB	768KB	768KB	768KB	1.536MB	1.792MB	2MB

4.1.2 Compiled RAM latency

The cache controller resets assume the slowest compiled RAMs are being used. This means eight cache controller clock cycles are used for each access. In terms of reads, the read data is sampled eight clock edges after the edge on which the RAM sampled the read request. Using this nomenclature, the shortest latency is one. This is the first latency example in Figure 4-1 on page 4-3. The latencies for each RAM are programmed in the Auxiliary Control Register as shown in *Auxiliary Control Register*

on page 2-9. See also *Miscellaneous signals* on page A-16 for descriptions of related signals. This register must only be programmed when the cache controller is not enabled.

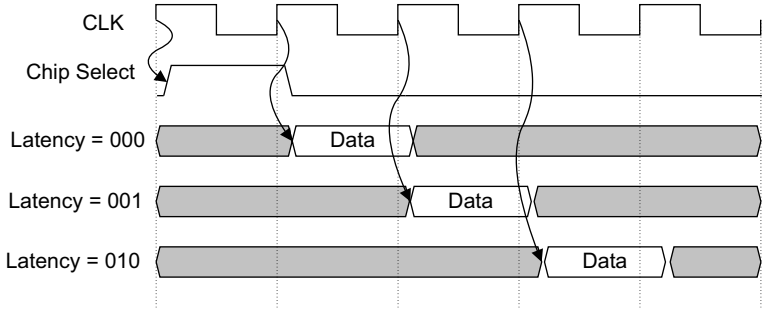


Figure 4-1 Compiled RAM latency

4.1.3 Eviction and linefill buffers

The eviction buffer for the cache controller can hold a full cache line or two half cache lines.This is emptied with a burst transfer.

There are two linefill buffers for the cache controller on M0 and M1. Slave ports can perform new read requests that hit in the cache, so that the slave ports are not blocked during linefills.

4.1.4 Cache Sync

The Cache Sync operation causes:

- Eviction buffer to be drained
- Write buffer to be drained
- Write allocate buffer to complete allocation.

The Cache Sync operation is considered as completed when all buffers (eviction, write, and write allocate) are empty, regardless of linefill activity on master read ports. So, an outstanding cache linefill can cause an eviction to occur after a Cache Sync operation completion.

A Cache Sync operation is always automatically performed when an atomic or background, cache maintenance operation is requested.

4.1.5 Asynchronous interface

The cache controller includes an asynchronous interface on the master ports. The behavior is similar to the asynchronous interface on ARM1136 processors. See the *ARM1136JF-S Technical Reference Manual* for details of clocking modes in ARM1136 processors. Table 4-2 shows the asynchronous mode control pins.

Table 4-2 Asynchronous mode control pins

Pins	Use
nRESET	Core reset
CLK	Core clock
HSYNCEN[a]	If **HSYNCEN** is HIGH, synchronous input is enabled. If **HSYNCEN** is LOW, asynchronous input is enabled.
HCLK	AHB clock
HCLKEN	Input
nHRESET	AHB reset

a. **HSYNCEN** does not exist if the asynchronous interface has been removed by synthesis.

The asynchronous interface logic can be removed through a synthesis option.

4.1.6 Asynchronous interface control

The use of the asynchronous interface is controlled by **HSYNCEN**. The logic for the asynchronous interface does not add any cycle delays when running in synchronous mode. Figure 4-2 on page 4-5 shows the read data timing logic for the asynchronous interface.

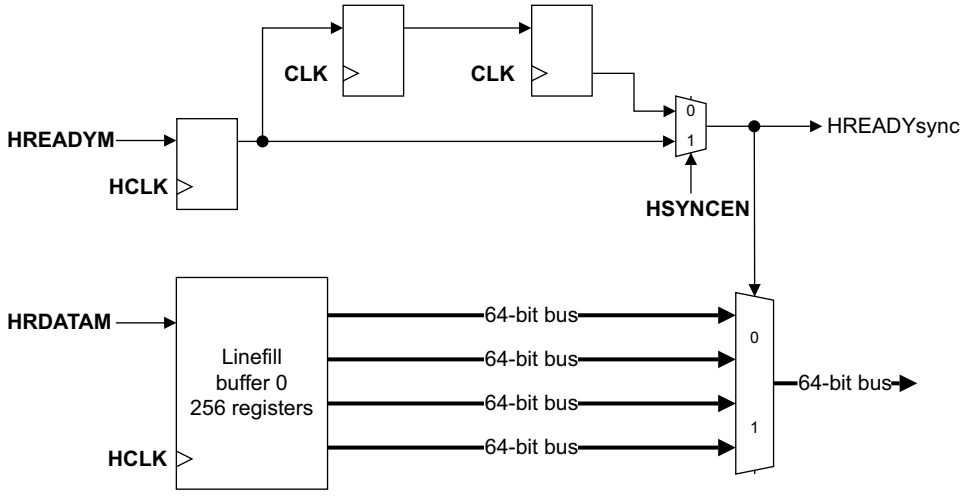


Figure 4-2 Asynchronous interface, read data

Figure 4-3 shows the write data timing logic for the asynchronous interface.

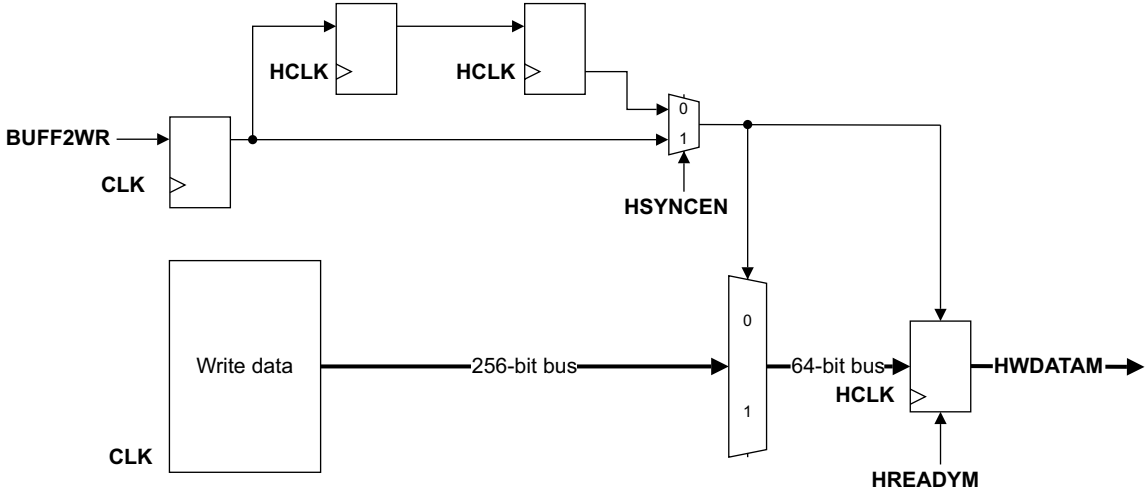


Figure 4-3 Asynchronous interface, write data

4.2 Clocking

The cache controller is clocked with the same clock as the CPU core.

The cache controller and master port interfaces can be clocked:

- synchronously with **CLK = HCLK**
- synchronously with **HCLK** as a submultiple of **CLK**
- asynchronously by using the asynchronous interface with **HSYNCEN** set LOW.

You must perform careful analysis to see which option is the most efficient.

All AMBA ports have their own clock enable to enable the transfer rate on the corresponding bus to be a submultiple of the port clock. Slave ports are clocked with **CLK** and have independent **CLKENSx**. Master ports are clocked with **HCLK** and have independent **HCLKENMx**.

4.3 Idle

The output signal IDLE indicates when the cache controller is not doing any internal processing. That is, all linefill buffers have been written to data RAM, the eviction buffer is empty, and there are no requests being processed. At this point, IDLE is asserted.

The write buffer may not be empty when the IDLE signal is set. If you assert IDLE to switch to wait for interrupt mode, drain the write buffer first.

If the L1 core has entered a wait for interrupt state, and now the cache controller is idle, the clock to both the L1 core and cache controller can be stopped. The cache controller must now be awoken at the same time or before the L1 core, so that any AHB requests from the L1 core are handled by the cache controller.

4.4 Disabled

When the cache controller block is present but not enabled, transactions are passed through to the L3 main memory system on the cache controller master port outputs. The penalty introduced by disabling the cache controller is twice the depth of internal cache controller pipeline, once for slave to master, and the other for master to slave.

4.5 Set-up sequence

After reset, the cache controller is disabled. You must configure the cache controller before use. This involves:

- setting up the cache controller registers for use
- invalidating the cache controller
- then enabling the cache controller.

This sequence properly configures the cache controller for use:

1. Configure way size and associativity.
2. Configure RAM latencies.
3. Perform an Invalidate by Way operation with all ways selected.
4. Turn the cache controller on when the Invalidate by Way completes.

4.6 Behavior for ARMv5 memory systems

The cache controller supports ARMv5 and ARM1136 AMBA extensions. This section describes ARMv5 memory system behavior:

- *ARMv5 system transactions* describes the transaction types
- *Behavior for ARMv5 transactions* on page 4-12 describes cache controller transactions on v5 systems.

4.6.1 ARMv5 system transactions

In a system that does not use the ARM1136 AMBA extensions, the cache controller slave port 2 is unused and the following pins must be tied off:

- **HBSTRBSx[7:0]** must be tied HIGH
- **HPROTSx[4]** must be tied to **HPROTSx[3]**
- **HPROTSx[5]** must be tied LOW
- **HUNALIGNSx** must be tied LOW
- **HRESPMx[2]** must be tied LOW.

You are provided with an additional piece of hardware for the master ports to emulate unaligned accesses in ARMv5 memory systems. This hardware block is equivalent to the Byte Lane Strobe Converter, BLScnv, available in the AMBA Design Kit, ADK. It removes the **HUNALIGN** and **HBSTRB** signals.

Note

The **HUNALIGNMx** and **HBSTRBMx** signals are only used in ARMv6 memory systems.

In the case of a 32-bit interface, read data and write data on slaves are treated as follows:

- read data is on **HRDATA[31:0]**, and **HRDATA[63:32]** is always 0.
- write data must be duplicated on **HWDATA[63:32]** and **HWDATA[31:0]**.

In the case of a 32-bit interface, read data and write data on masters are treated as follows:

- write data is on **HWDATA[31:0]**, and **HWDATA[63:32]** is always 0.
- read data must be duplicated on **HRDATA[63:32]** and **HRDATA[31:0]**.

Table 4-3 shows the **HPROTSx[4:2]** and *Translation Lookaside Buffer* (TLB) attribute correspondences in an ARMv5 system.

Table 4-3 HPROTSx[4:2] and TLB correspondences in an ARMv5 system

HPROT[4] tied to HPROT[3]	HPROT[3]	HPROT[2]	ARMv5 system
0	0	0	**NCNB** No buffered writes, no linefills
0	0	1	**NCB** Buffered writes, no linefills
1	1	0	**WT** Buffered writes, linefills on reads, but not writes
1	1	1	**WB** Buffered writes on misses, linefills on reads, but not writes

Write allocate override causes all cacheable writes misses to be allocated to the cache controller. See *Auxiliary Control Register* on page 2-9 for a description of the write allocate override bit.

4.6.2 Behavior for ARMv5 transactions

Table 4-4 shows cache controller behavior for ARMv5 transactions.

Table 4-4 Behavior for ARMv5 transactions

ARMv5 memory region	Behavior
NCNB	Read Not cached in the L2, causes AHB master port access. Write Not buffered, causes AHB master port access.
NCB	Read Not cached in the L2, causes AHB master port access. Write Not buffered, causes AHB master port access.
Cached **WT** mode	Read hit Read from the L2. Read miss Linefill to the L2, then forwarded to L1. Write hit Put in write buffer, write to the L2 when write buffer drained and causes AHB master port access when write buffer drained. Write miss Put in write buffer, causes AHB master port access when write buffer drained.
Cached WB mode	Read hit Read from the L2. Read miss Linefill to the L2, then forwarded to L1. Write hit Put in write buffer, write to the L2 when write buffer drained, mark line as dirty. Write miss Put in write buffer, causes AHB master port access when write buffer drained.

Write allocate override causes all cacheable writes misses to be allocated to the cache controller. See *Auxiliary Control Register* on page 2-9 for a description of the write allocate override bit.

The write buffer is drained when the two slots contain data, the least used line is drained first, or in the case of hazards the line causing the hazard is drained. Refer to *Write buffer* on page 2-23 for details about the write buffer.

Unaligned burst accesses and known length bursts

Unaligned burst accesses are not supported. If a noncacheable known length burst is early terminated on a slave port, the behavior of the master port handling the transaction is unpredictable.

4.7 Behavior for ARMv6 memory systems

This section describes cache controller behavior for ARMv6 memory systems:

- *ARMv6 system transactions*
- *Behavior for ARMv6 transactions* on page 4-15

4.7.1 ARMv6 system transactions

In the case of a 32-bit interface, read data and write data on masters are treated as follows:

- write data is on **HWDATA[31:0]**, and **HWDATA[63:32]** is always 0.
- read data must be duplicated on **HRDATA[63:32]** and **HRDATA[31:0]**.

Table 4-5 shows the **HPROTSx[4-2]** and TLB correspondences in ARMv6.

Table 4-5 HPROTSx[4-2] and TLB correspondences in ARMv6

HSIDEBAND[1] Shared	HPROT[4] Allocate	HPROT[3] cacheable	HPROT[2] Bufferable	ARMv6 ARM1136 AMBA extensions	ARMv5 equivalent
-	0	0	0	Strongly ordered, always Shared No buffered writes, no linefills.	NCNB
-	0	0	1	Device, always Shared No linefills.	NCB
0	0	1	0	Outer noncacheable, nonshared No buffered writes, no linefills.	-
1	0	1	0	Outer noncacheable, shared No buffered writes, no linefills.	-
0	0	1	1	Outer Write-Back write-allocate (OWBWA) – nonshared Buffered writes on misses, linefills on reads and writes.	-
1	0	1	1	Outer Write-Back write-allocate (OWBWA) – Shared Buffered writes on misses, no linefills	-
-	1	0	-	N/A	-

Table 4-5 HPROTSx[4-2] and TLB correspondences in ARMv6 (continued)

HSIDEBAND[1] Shared	HPROT[4] Allocate	HPROT[3] cacheable	HPROT[2] Bufferable	ARMv6 ARM1136 AMBA extensions	ARMv5 equivalent
0	1	1	0	Outer Write-Through no-write-allocate (OWTNWA) – nonshared Buffered writes, linefills on reads but not writes.	WT
1	1	1	0	Outer Write-Through no-write-allocate (OWTNWA) – Shared Buffered writes, no linefills	-
0	1	1	1	Outer Write-Back no-write-allocate (OWBNWA) – nonshared Buffered writes on misses, linefills on reads but no writes	WB
1	1	1	1	Outer Write-Back no-write-allocate (OWBNWA) – Shared Buffered writes on misses, no linefills	-

4.7.2 Behavior for ARMv6 transactions

Table 4-6 shows memory types and the cache controller behavior for reads and writes.

Table 4-6 Behavior with ARMv6 memory types

Memory type	Reads	Writes
Strongly Ordered (SO), Always Shared	Not cached in the L2. Causes AHB master port access.	Not buffered. Causes AHB master port access.
Device (DV), Always Shared	Not cached in the L2. Causes AHB master port access.	Not buffered. Causes AHB master port access.
Outer noncacheable (ONC), Nonshared, (Same as SO)	Not cached in the L2. Causes AHB master port access.	Not buffered. Causes AHB master port access.
Outer noncacheable (ONC), Shared, (Same as SO)	Not cached in the L2. Causes AHB master port access.	Not buffered. Causes AHB master port access.

Table 4-6 Behavior with ARMv6 memory types (continued)

Memory type	Reads	Writes
Outer Write-Back write-allocate (OWBWA), nonshared	Read hit: Read from the L2. Read miss: Linefill to the L2.	Write hit Put in write buffer, write to the L2 when write buffer drained, and mark line as dirty. Write miss Put in write buffer, put in write-allocate buffer when write buffer drained, data request to L3 if line is not full, allocation to L2.
Outer Write-Back write-allocate (OWBWA), Shared	Not cached in the L2. Causes AHB master port access.	Put in write buffer. Write to L3 when write buffer drained.
Outer Write-Through no-write-allocate (OWTNWA), nonshared	Read hit: Read from the L2. Read miss: Linefill to the L2.	Write hit Put in write buffer, write to L2, causes AHB master port access when write buffer drained. Write miss Put in write buffer, causes AHB master port access when write buffer drained.
Outer Write-Through no-write-allocate (OWTNWA), Shared	Not cached in the L2 Causes AHB master port access.	Put in write buffer, causes AHB master port access when write buffer drained.
Outer Write-Back no-write-allocate (OWBNWA), nonshared	Read hit: Read from the L2. Read miss: Linefill to the L2.	Write hit: Put in write buffer, write to the cache controller when write buffer drained, mark line as dirty. Write miss: put in write buffer, causes AHB master port access when write buffer drained.
Outer Write-Back no-write-allocate (OWBNWA), Shared	Not cached in the L2 Causes AHB master port access.	Put in write buffer, causes AHB master port access when write buffer drained.

Note

- ARMv6 specifies that accesses from a processor to memory marked as Device must occur at the size and the order defined by the instruction. So, Device write accesses are treated by cache controller as nonbufferable because the write buffer is a merging write buffer.

- Write allocate override causes all nonshared cacheable write misses, unless shared attribute override is set, to be allocated to the cache controller. Setting the shared attribute override bit in the Auxiliary Control Register causes a Shared access to be internally treated as NonShared in the cache controller. See *Auxiliary Control Register* on page 2-9 for more information on the write allocate override bit and the shared attribute override bit.
- No unexpected hits can occur on noncacheable accesses. Slave ports request a transfer to the internal cache controller or to the master port depending on access attributes. This means that no lookups are performed for noncacheable accesses.

4.7.3 HBSTRBSx and HUNALIGNSx

HBSTRBSx signals indicate which byte lanes are active for the current transaction. One **HBSTRBSx** signal is used for each eight bits of the data bus. The **HBSTRBSx** signals are asserted in the same cycles as the other address and control signals of the transfer to which it applies. In other words, it is an address phase signal.

HADDRSx and **HSIZESx** are used to define the container in which the **HBSTRBSx** signals can be active. The size of the transaction is sufficient to cover all of the bytes being written and covers more bytes in the case of a misaligned transfer. For an ARM1136 core **HADDRSx** is aligned to the size of the transfer, as indicated by **HSIZESx**, so that the address of the transfer is rounded down to the nearest boundary of the size of the transaction.

Note

Byte strobes are required for both read and write transfers. For ARMv5 systems, all **HBSTRBSx** signals must be tied HIGH, with all **HUNALIGNSx** signals tied LOW, **HADDRSx** and **HSIZESx** generate the internal byte lane signals.

HUNALIGNSx signals are for an ARMv6 system and must be provided by masters that can produce unaligned accesses. This signal is only to assist with backward compatibility and indicates when a single unaligned transfer occurs that requires more than one AHB 2.0 transfer, without byte strobes.

The **HUNALIGNSx** signals are address phase signals and must be asserted HIGH for unaligned transfers, and LOW for aligned transfers. Table 4-7 shows example uses of **HBSTRBSx** and **HUNALIGNSx** in little-endian mode.

Table 4-7 Example uses of HBSTRBSx and HUNALIGNSx, little-endian

Transfer Description	HADDRSx	HSIZESx	HBSTRBSx	HUNALIGNSx
8-bit access to 0x1000	0x1000	0x0	b00000001	0
8-bit access to 0x1003	0x1003	0x0	b00001000	0
8-bit access to 0x1007	0x1007	0x0	b10000000	0
16-bit access to 0x1000	0x1000	0x1	b00000011	0
16-bit access to 0x1005	0x1004	0x2	b01100000	1
16-bit access to 0x1007	0x1007	0x0	b10000000	0
	0x1008	0x0	b00000001	0
32-bit access to 0x1000	0x1000	0x2	b00001111	0
32-bit access to 0x1002	0x1000	0x3	b00111100	1
32-bit access to 0x1003	0x1000	0x3	b01111000	1
32-bit access to 0x1007	0x1007	0x0	b10000000	0
	0x1008	0x2	b00000111	1
64-bit access to 0x1000	0x1000	0x3	b11111111	0
64-bit access to 0x1003	0x1000	0x3	b11111000	1
	0x1008	0x2	b00000111	1

Unaligned burst accesses and known length bursts

Unaligned burst accesses are not supported. If a noncacheable known-length burst is early terminated on a slave port, the behavior of the master port handling the transaction is unpredictable.

4.7.4 Locked and exclusive access

Locked and exclusive accesses are treated as special cases:

- *Locked accesses* on page 4-19
- *Exclusive accesses* on page 4-19.

Locked accesses

Locked accesses use **HPROT** information. In the case of Noncacheable transfers, Strongly Ordered, Device, Normal Shared and Normal NonShared Outer Noncacheable, the access is forwarded to L3 through master ports and is marked as locked. No cache lookups are performed in that case.

In the case of cacheable transfers, Normal NonShared or Normal Shared with shared attribute override, (excepting Outer noncacheable for both cases), lock information is ignored. A cache lookup is always performed and, in the case of a cache miss, a linefill, nonlocked, is requested on the master side.

Write accesses (Write-Through or Write-Back miss) cause nonlocked writes on the master side.

Exclusive accesses

An exclusive access, always Shared out of ARM1136 cores, is always forwarded directly to master ports without any lookup in the cache because Shared memory regions are treated as noncacheable regions. A region of memory marked as Shared is one in which the effect of interposing a cache, or caches, on the memory system is entirely transparent. Implementations can use a variety of mechanisms to support this, from not caching accesses in Shared regions to more complex hardware schemes for cache coherency for those regions. By default, the cache controller does not cache accesses to Shared regions.

By default, the cache controller does not support exclusive access to memory regions marked as Shared Outer-Cacheable with shared override bit set. The shared override bit locally allows the cache controller to treat Shared regions as NonShared. Because the Shared attribute is a ARM v6 global attribute, this means the Inner region in ARM1136J(F)-S processors is noncached, and cached in the cache controller. This means an exclusive access would not reach L3 if it hit in the cache controller. The shared override bit opens all sorts of possibilities for error with exclusive accesses. As the cache controller does not implement an exclusive monitor for its default behavior, it sends an error response if an exclusive access is attempted with the shared override bit set to a Shared Outer-Cacheable region. The abort disable bit, bit[24] of the Auxiliary Control Register, offers the option to change that default behavior to enable the case of a single-processor system with an exclusive monitor inserted between the core and the cache controller.

For exclusive accesses, the exclusive access monitor is not implemented in the cache controller. The exclusive access monitor must be implemented at the level where arbitration for multiple masters is performed.

4.8 Timing diagrams

This section describes the timings of typical cache controller operations. It contains the following sections:

- *Hit*
- *Miss*
- *Two simultaneous hits* on page 4-21
- *Buffered write* on page 4-22
- *Outer noncacheable access* on page 4-23.

4.8.1 Hit

Figure 4-4 shows the timings of operations for a cache controller hit.

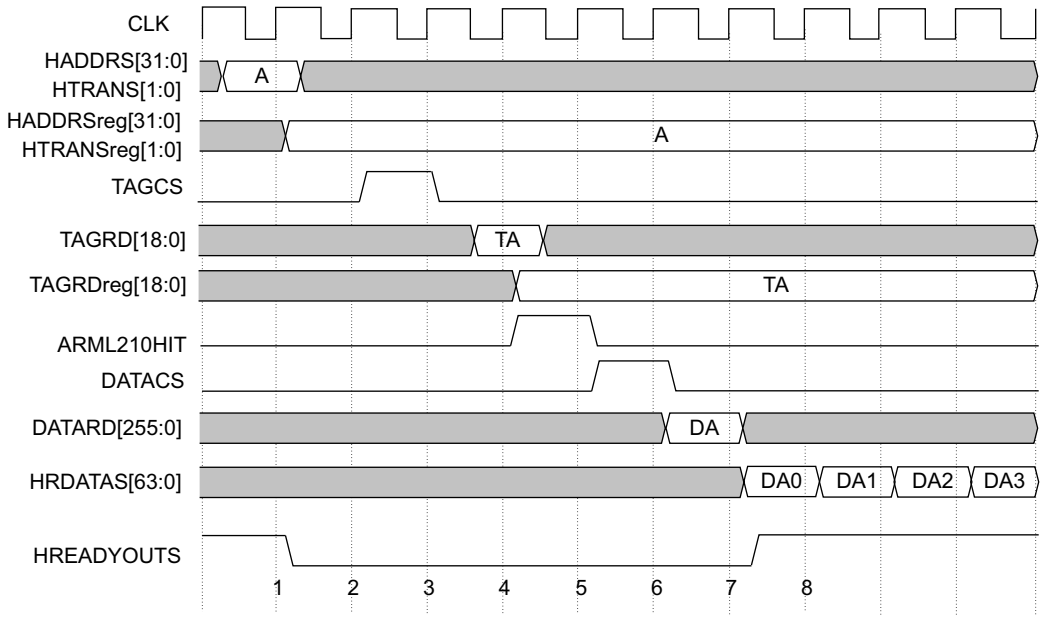


Figure 4-4 Hit

4.8.2 Miss

Figure 4-5 on page 4-21 shows the timings of cache controller operations for a cache controller miss.

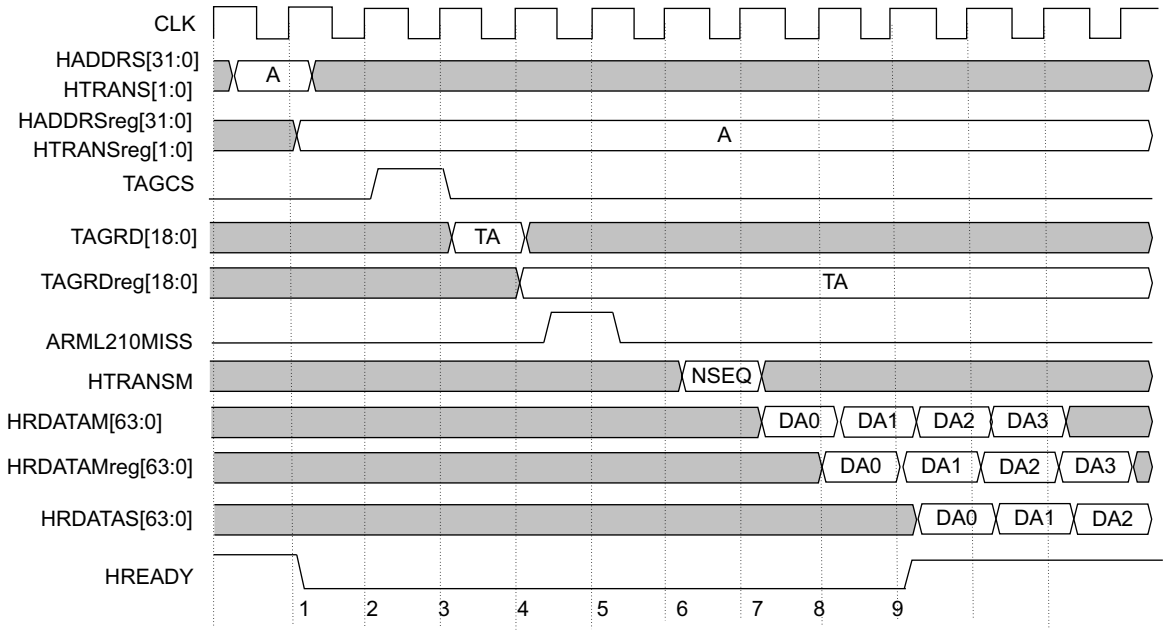


Figure 4-5 Miss

4.8.3 Two simultaneous hits

Figure 4-4 on page 4-20 shows the timings of cache controller operations for two simultaneous hits.

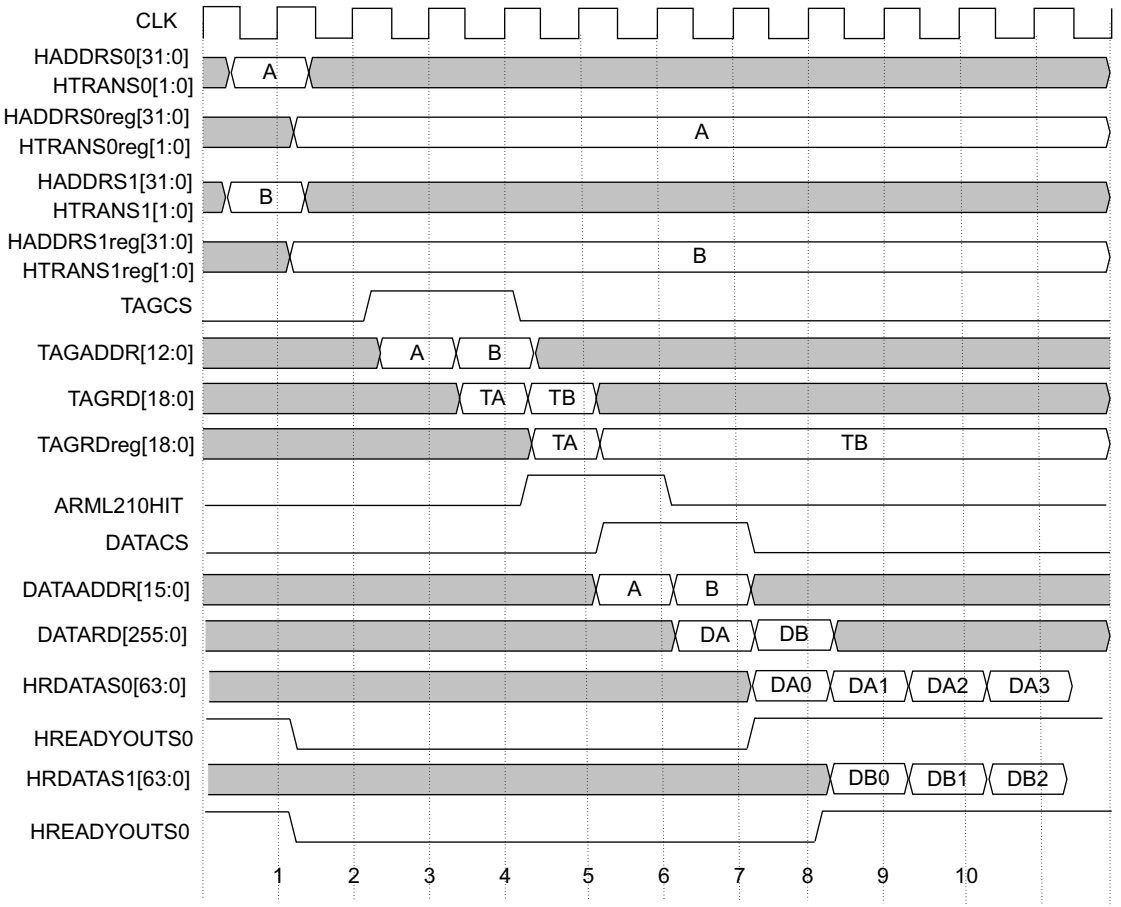


Figure 4-6 Simultaneous hits

4.8.4 Buffered write

Figure 4-4 on page 4-20 shows the timings of cache controller operations for a buffered write.

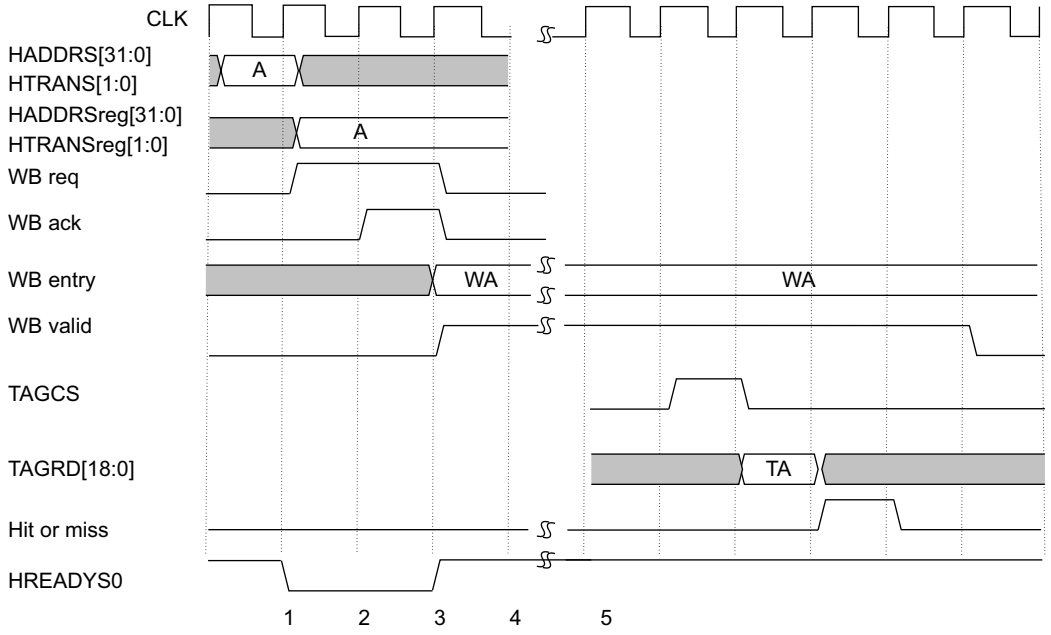


Figure 4-7 Buffered write timings

4.8.5 Outer noncacheable access

Figure 4-8 on page 4-24 shows the timings of cache controller operations for an outer noncacheable access.

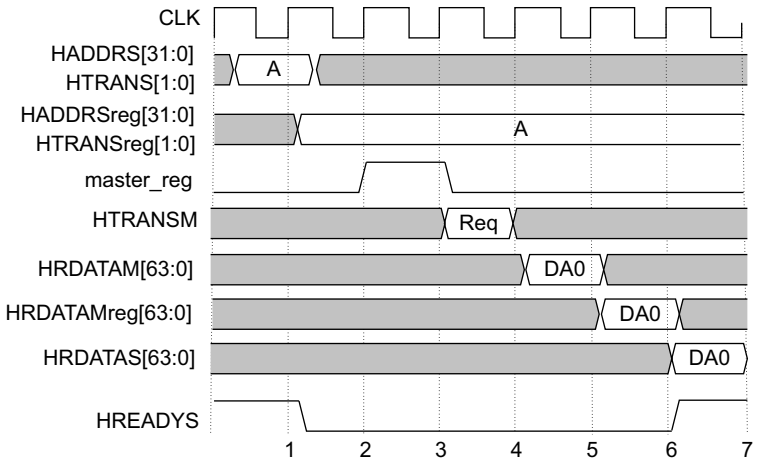


Figure 4-8 Outer noncacheable access

Chapter 5
Design for Test

This chapter describes the cache controller design for test features. It contains the following sections:

- *About design for test* on page 5-2
- *Memory built-in self-test, MBIST* on page 5-3.

5.1 About design for test

To guarantee signal integrity with the AHB and RAM interfaces, all inputs and outputs of the cache controller are registered, with these exceptions:

- **nRESET, nHRESET**
- **CLKENS0**, **CLKENS1**, and **CLKENS2**
- **HCLKENM0**, **HCLKENM1**, and **HCLKENM2**
- **HREADYM0**, **HREADYM1**, **HREADYM2**, **HREADYS0**, **HREADYS1**, and **HREADYS2**
- **HRESPS0, HRESPS1, and HRESP2**
- **DATARD, DATAPRD, and DATAERR**.

This simplifies scan testing of the block, and full scan testing is supported.

5.2 Memory built-in self-test, MBIST

MBIST tests the memory cells of the compiled level 2 memory RAMs. MBIST writes and reads all locations of the RAM to ensure that the cells are operating correctly. This process gives additional test coverage of the address and data paths that MBIST uses.

MBIST mode takes priority over all other modes. When MBIST mode is activated with the **MTESTON** pin, the L2 RAMs are accessible only to the cache controller MBIST block. In functional mode **MTESTON** must be kept LOW. Only one RAM can be accessed by the MBIST port at a time. Figure 5-1 on page 5-4 shows the cache controller MBIST configuration.

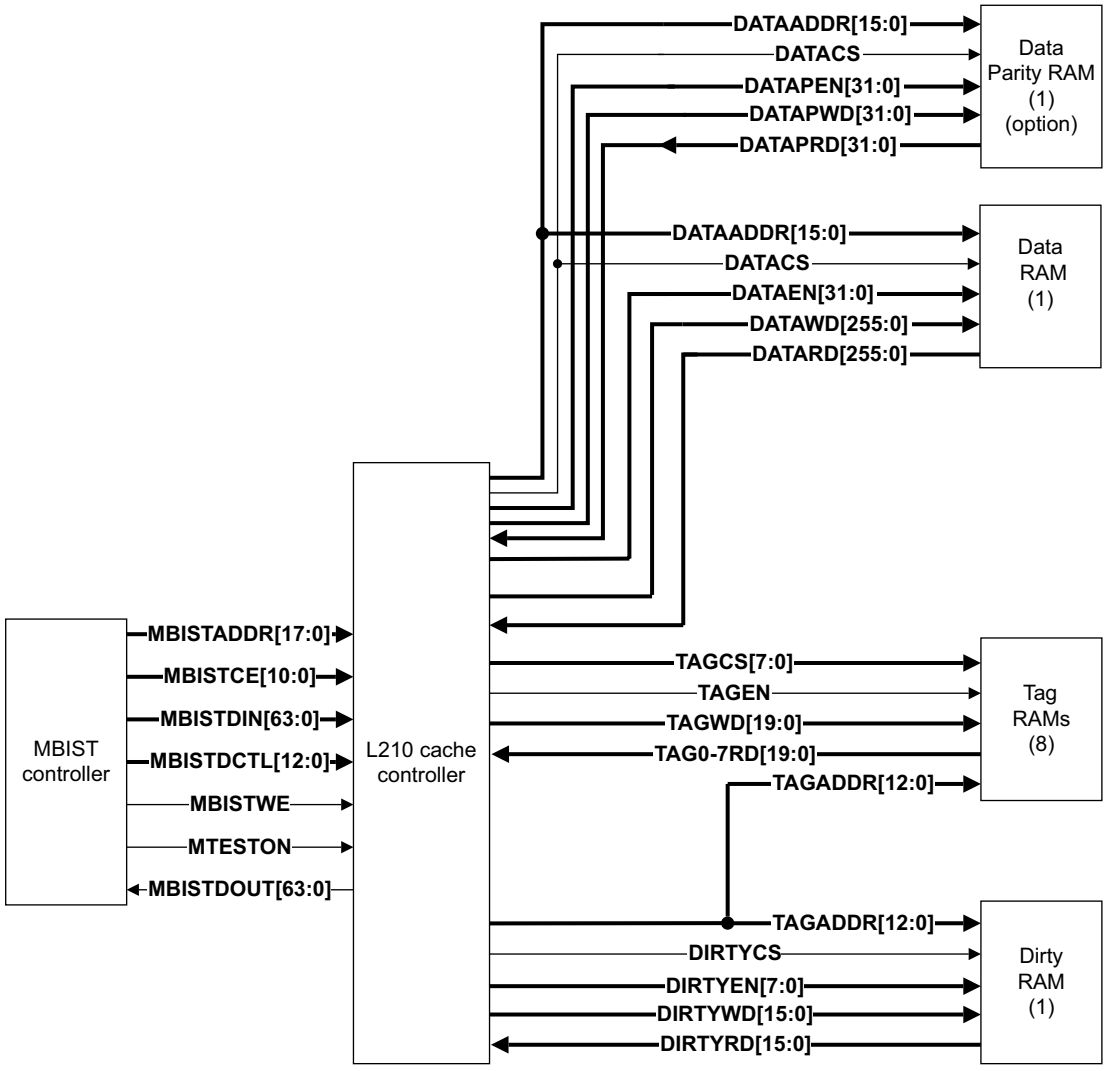


Figure 5-1 MBIST configuration

Partners can choose to design their own MBIST controller. Alternatively, ARM has an MBIST controller available for use.

5.2.1 MBIST compiled RAM latencies

During functional operation, the cache controller resets assuming that the slowest compiled RAMs are being used, and the Auxiliary Control Register is used to configure the latency. This is not the case during MBIST operation. When placing the cache controller into MBIST mode by setting the **MTESTON** signal HIGH, the effects of the Auxiliary Control Register are overridden, and the MBIST pipeline is statically configured for zero additional cycles of latency. It is the responsibility of the MBIST controller to insert any required wait states when testing RAMs that require additional cycles of latency.

See *Auxiliary Control Register* on page 2-9 and *Compiled RAM latency* on page 4-2 for details.

For MBIST, the user must know the latencies of the RAMs being tested, and the MBIST block must be designed to support them.

5.2.2 MBIST testing of cache controller data RAM

The cache controller data RAM is 256 bits wide, and the size of the MBISTDIN and MBISTDOUT buses on the cache controller MBIST interface is 64 bits, so four reads and four writes are required for each index of the data RAM. When writing to the RAM, **MBISTADDR[1:0**] is used as a doubleword select for each index. Table 5-1 shows this.

Table 5-1 Using MBISTADDR as an index for data RAM writes

MBISTADDR[1:0]	DATAEN[31:0]	DATAWD used
b00	0x000F	[63:0]
b01	0x00F0	[127:64]
b10	0x0F00	[191:128]
b11	0xF000	[255:192]

For reads from a previous MBIST transaction, **MBISTDCTL[1:0]** is used. The state of **MBISTADDR[1:0]** and **MBISTDCTL[1:0]** is only important when accessing the Data RAM. For all other RAMs, these signals can have any value without corrupting

the output data. Separate pins are needed because the MBIST transactions are pipelined. Table 5-2 shows the address range of **MBISTADDR** used to test the data RAM, based on the cache controller size.

Table 5-2 MBISTADDR and MBISTDIN mappings for data RAM

Size	Number of data RAM indexes	MBISTADDR to data RAM mapping	MBISTDIN to data RAM mapping
128KB	4096	**DATAADDR[11:0] = MBISTADDR[17:15,10:2]**	**DATAWD[63:0] = MBISTDIN[63:0]**
256KB	8192	**DATAADDR[12:0] = MBISTADDR[17:15,11:2]**	**DATAWD[63:0] = MBISTDIN[63:0]**
512KB	16384	**DATAADDR[13:0] = MBISTADDR[17:15,12:2]**	**DATAWD[63:0] = MBISTDIN[63:0]**
1MB	32768	**DATAADDR[14:0] = MBISTADDR[17:15,13:2]**	**DATAWD[63:0] = MBISTDIN[63:0]**
2MB	65536	**DATAADDR[15:0] = MBISTADDR[17:2]**	**DATAWD[63:0] = MBISTDIN[63:0]**

The cache controller has 2 line read buffers (LRB), each 256 bits wide. One of these, LRB1, holds data for MBIST testing. The cache controller always adds 2 register delays to the MBIST data read path for the data RAM.

The cache controller MBIST block must account for the data RAM latency in the pipeline. The latency can be from one-eight clock cycles as described in *MBIST compiled RAM latencies* on page 5-5.

Figure 5-2 on page 5-7 shows the MBIST data RAM test paths. In Figure 5-2 on page 5-7 **MBISTCE[0]** is for the chip enable to the data RAM. **MBISTDCTL[2:0]** is for reads from previous MBIST transactions with **MBISTDCTL[2]** set to 1 and **MBISTDCTL[12:3]** set to 0 to select the data RAM read data on the cycle that the read data is valid at the output of the RAM.

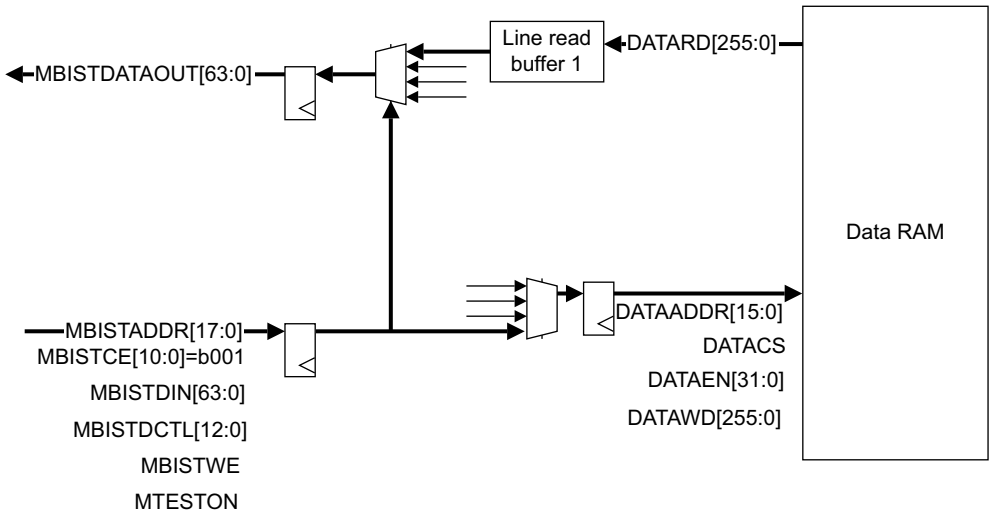


Figure 5-2 MBIST Data RAM test paths

5.2.3 MBIST testing of cache controller data parity RAM

There is one data parity RAM associated with the cache controller. Table 5-3 shows the address ranges of **MBISTADDR** and **MBISTDIN** used to test the data parity RAM, based on the cache controller size.

Table 5-3 MBISTADDR and MBISTDIN mappings for data parity RAM

Size	Number of RAM indexes	MBISTADDR to RAM mapping	MBISTDIN to RAM mapping
128KB	4096	**DATAADDR[11:0] = MBISTADDR[17:15,10:2]**	**DATAPWD[31:0] = MBISTDIN[31:0]**
256KB	8192	**DATAADDR[12:0] = MBISTADDR[17:15,11:2]**	**DATAPWD[31:0] = MBISTDIN[31:0]**
512KB	16384	**DATAADDR[13:0] = MBISTADDR[17:15,12:2]**	**DATAPWD[31:0] = MBISTDIN[31:0]**
1MB	32768	**DATAADDR[14:0] = MBISTADDR[17:15,13:2]**	**DATAPWD[31:0] = MBISTDIN[31:0]**
2MB	65536	**DATAADDR[15:0] = MBISTADDR[17:2]**	**DATAPWD[31:0] = MBISTDIN[31:0]**

The cache controller has two line read buffers (LRB), each 256 bits wide. One of these, LRB1, holds data for MBIST testing. The cache controller always adds 2 register delays to the MBIST data read path for the data parity RAM.

The cache controller MBIST block must account for the data parity RAM latency in the pipeline. The latency can be from one to eight clock cycles. *MBIST compiled RAM latencies* on page 5-5 describes this.

Figure 5-3 shows the MBIST paths for data parity RAM testing. In Figure 5-3 **MBISTCE[10]** is for the chip enable to the data parity RAM. **MBISTDCTL[12]** is for reads from previous MBIST transactions. **MBISTDCTL[12]** must be set to 1 and **MBISTDCTL[11:2]** must be set to 0 to select the Data parity RAM read data during the same cycle that the read data is valid at the output of the RAM.

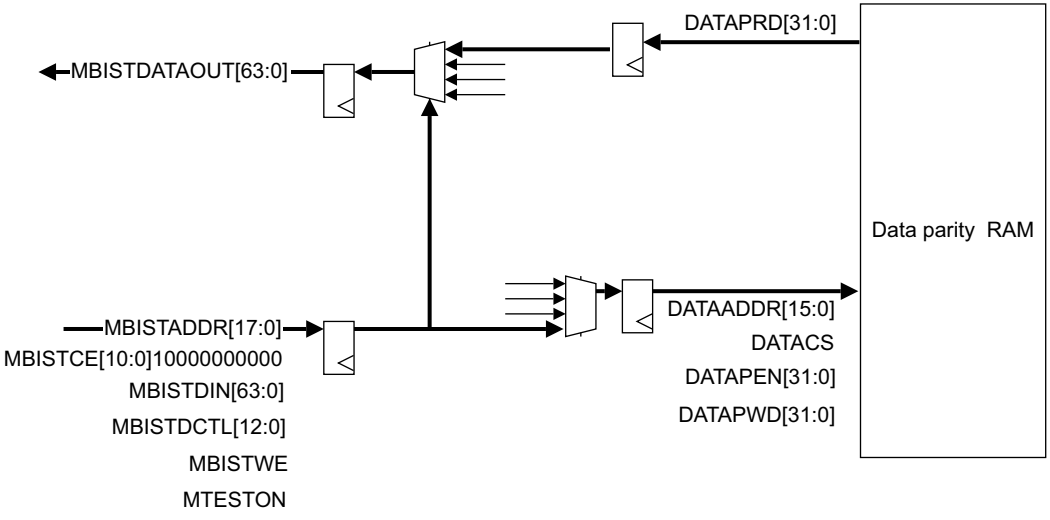


Figure 5-3 Data parity RAM for test

5.2.4 MBIST testing of the cache controller tag RAMs

There is one tag RAM for each way of the cache controller. The maximum number of tag RAMs the MBIST controller must test is eight. Only one tag RAM is tested at a time. Table 5-4 shows the address range of **MBISTADDR** used to test a tag RAM, based on the cache controller size. The parity bit for each TAG RAM present is tested along with the rest of the TAG array and is mapped to **MBISTDIN[19]**.

Table 5-4 MBISTADDR and MBISTDIN mapping for Tag RAMs

Size	Number of tag RAM indexes	MBISTADDR to RAM mapping	MBISTDIN to RAM mapping
128KB	512	**TAGADDR[8:0] = MBISTADDR[10:2]**	**TAGWD[19:0] = MBISTDIN[18:0, 19]**
256KB	1024	**TAGADDR[9:0] = MBISTADDR[11:2]**	**TAGWD[18:0] = MBISTDIN[18:1, 19]**
512KB	2048	**TAGADDR[10:0] = MBISTADDR[12:2]**	**TAGWD[17:0] = MBISTDIN[18:2, 19]**
1MB	4096	**TAGADDR[11:0] = MBISTADDR[13:2]**	**TAGWD[16:0] = MBISTDIN[18:3, 19]**
2MB	8192	**TAGADDR[12:0] = MBISTADDR[14:2]**	**TAGWD[15:0] = MBISTDIN[18:4, 19]**

The data from the tag RAMs is always registered by the cache controller. In addition, there is a register on the MBIST port of the cache controller. The cache controller always adds two register delays to the MBIST data read path for the tag RAMs. The latency of the tag RAMs can be from one to eight clock cycles as described in *MBIST compiled RAM latencies* on page 5-5.

Figure 5-4 on page 5-10 shows the MBIST paths for tag RAM testing. In Figure 5-4 on page 5-10 **MBISTCE[8:1]** is for chip enables to the tag RAMs. **MBISTCE[8:1]** corresponds to **TAGCS[7:0]**. **MBISTDCTL[10:3]** is for reads from previous MBIST transactions, which controls the read data for Tag RAMs 7-0 respectively. The **MBISTDCTL** bit corresponding to the required RAM read data must be asserted on the same cycle that the read data is available at the output of the RAM, and all other bits of **MBISTDCTL**, except **MBISTDCTL[1:0]**, must be kept LOW. Only bits[19:0] of **MBISTDIN** and **MBISTDOUT** are used.

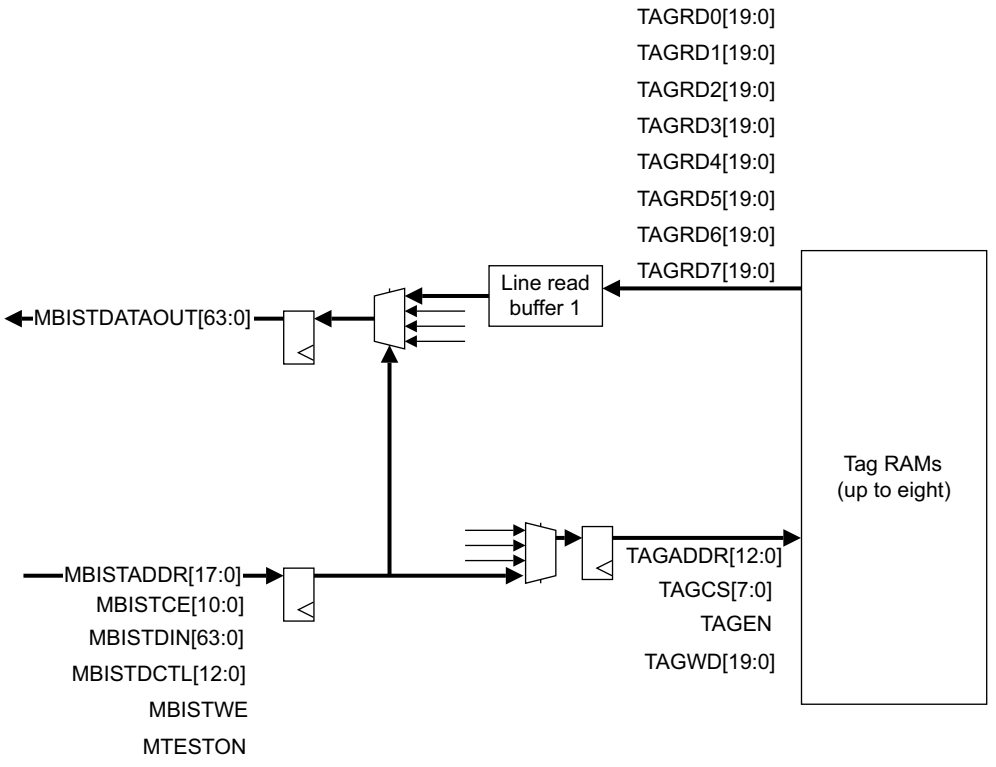


Figure 5-4 MBIST paths for tag RAM testing

5.2.5 MBIST testing of the cache controller dirty RAM

There is one dirty RAM associated with the cache controller. The dirty RAM uses the same address as the tag RAMs. Table 5-5 shows the address range of **MBISTADDR** used to test the dirty RAM, based on the cache controller size.

Table 5-5 MBISTADDR and MBISTIDIN mappings for dirty RAM

Size	Number of dirty RAM Indexes	MBISTADDR to dirty RAM mapping	MBISTDIN to dirty RAM mapping
128KB	512	**TAGADDR[8:0] = MBISTADDR[10:2]**	**DIRTYWD[15:0] = MBISTDIN[15:0]**
256KB	1024	**TAGADDR[9:0] = MBISTADDR[11:2]**	**DIRTYWD[15:0] = MBISTDIN[15:0]**

Table 5-5 MBISTADDR and MBISTIDIN mappings for dirty RAM (continued)

Size	Number of dirty RAM Indexes	MBISTADDR to dirty RAM mapping	MBISTDIN to dirty RAM mapping
512KB	2048	**TAGADDR[10:0] = MBISTADDR[12:2]**	**DIRTYWD[15:0] = MBISTDIN[15:0]**
1MB	4096	**TAGADDR[11:0] = MBISTADDR[13:2]**	**DIRTYWD[15:0] = MBISTDIN[15:0]**
2MB	8192	**TAGADDR[12:0] = MBISTADDR[14:2]**	**DIRTYWD[15:0] = MBISTDIN[15:0]**

The data from the dirty RAM is always registered by the cache controller. In addition, there is a register on the MBIST port of the cache controller. So, to the cache controller MBIST controller, the cache controller always adds two register delays to the MBIST data read path for the dirty RAM. The latency can be from one to eight clock cycles as described in *MBIST compiled RAM latencies* on page 5-5.

Figure 5-5 shows the paths for dirty RAM testing using MBIST.

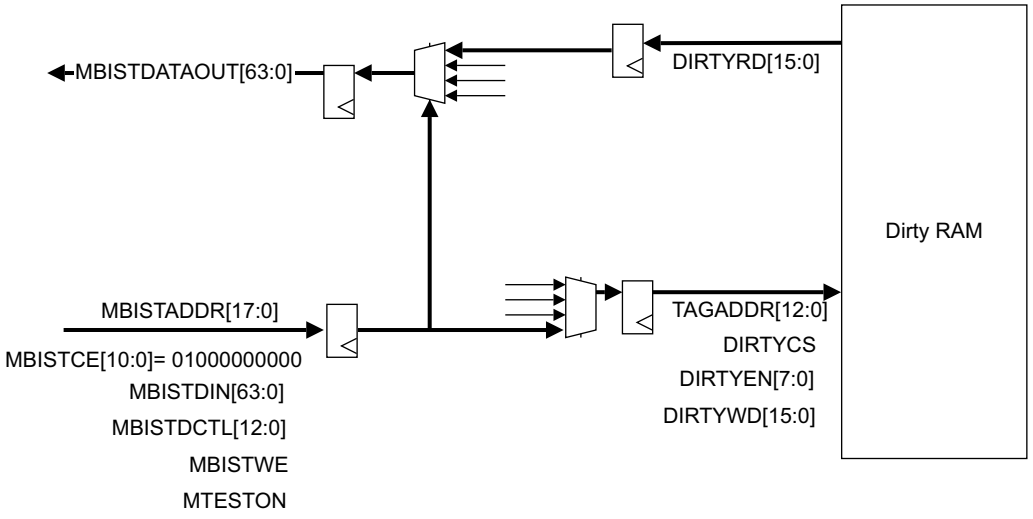


Figure 5-5 Dirty RAM testing paths using MBIST

In Figure 5-5 **MBISTCE[9]** is for the write enable to the dirty RAM. **MBISTCE[9]** corresponds to **DIRTYCS**. **MBISTDCTL[11]** is for reads from previous MBIST transactions. **MBISTDCTL[11]** must be set to 1 and **MBISTDCTL[12, 10:2]** must be set to 0 to select the dirty RAM read data on the cycle that the read data is valid at the output of the RAM. The dirty RAM uses the same address as the tag RAM.

Chapter 6
Parity and RAM Error Support

This chapter describes the support for parity errors in the data and tag RAMs. It contains the following sections:

- *Parity and RAM error support* on page 6-2
- *Error signaling and handling* on page 6-4.

6.1 Parity and RAM error support

The cache controller generates parity write data for the data and tag RAMs as shown in Figure 6-1 on page 6-3. For the data RAM, the cache controller generates parity write data on a per-byte basis. For the tag RAMs, the cache controller generates one parity bit that must be routed to all tag RAMs. Only one tag RAM is written at any one time so only one bit is required. The generation and checking of parity data is disabled if bit 21 of the Auxiliary Control Register is set to 0 as described in *Auxiliary Control Register* on page 2-9.

For AHB read transactions, if a parity error occurs on tag or data RAM, an error is reported back to **HRESPSx** and through the event bus.

For AHB write transactions and cache maintenance operations, if a parity error occurs on tag or data RAM, tag RAM only for AHB write transactions, the error is reported back through the event bus only.

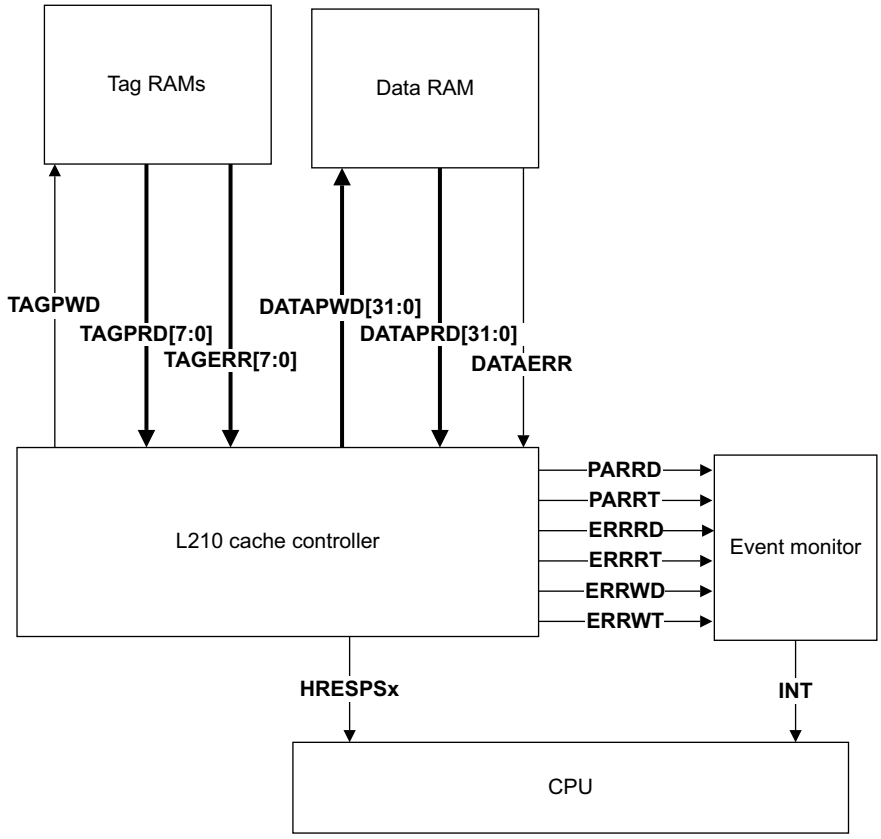


Figure 6-1 Parity and RAM error support

6.2 Error signaling and handling

Parity and RAM errors are reported to the CPU through the event monitor or the **HRESPSx** signals depending on the fault and the transaction that causes the fault. All cases and their effects are described as follows:

Cacheable read request on S0 or S1 ports

In the case of tag parity or RAM errors on cache lookup the event is flagged through **PARRT** or **ERRRT** to the event monitor. All accesses that are part of the request receive an ERROR response.

RDATASx values are Unpredictable.

In the case of data parity or RAM errors on data read the event is flagged through **PARRD** or **ERRRD** to the event monitor. All accesses that are part of the request receive an ERROR response.

RDATASx values are those read from Data RAM.

Write-Through or Write-Back write access from write buffer

In the case of tag parity or RAM error on cache lookup the event is flagged through **PARRT** or **ERRRT** to the event monitor. No attempt is made to write the cache.

In the case of data RAM error on data write the event is flagged through **ERRWD** to the event monitor. The cache line is marked as dirty in cases of write-back access. Subsequent reads to the same line are Unpredictable.

Allocation request from M0 or M1 masters or write-allocate buffer

In the case of a tag parity error or RAM error on cache lookup, for next victim selection, the event is flagged through **PARRT** or **ERRRT** to the event monitor. The allocation process is not stopped.

In the case of data parity error or RAM error on data read for eviction (all ways contain data to targeted index and next victim line is dirty), the event is flagged through **PARRD** or **ERRRD** to the event monitor. The allocation process is not stopped but the eviction is not performed

In the case of a tag RAM error on tag write the event is flagged through **ERRWT** to the event monitor. Subsequent cache lookups to the same index are Unpredictable.

In the case of data RAM error on a data write the event is flagged through **ERRWD** to the event monitor. Subsequent reads to that line are Unpredictable.

Clean maintenance operation

In the case of tag parity or RAM error on cache lookup the event is flagged through **PARRT** or **ERRRT** to the event monitor. The clean operation is canceled.

In the case of data parity error or RAM error on a data read the event is flagged through **PARRD** or **ERRRD** to the event monitor. The clean operation is canceled.

Invalidate maintenance operation:

In the case of tag parity error or RAM error on cache lookup, for an Invalidate by Address operation, the event is flagged through **PARRT** or **ERRRT** to the event monitor. The invalidate operation is canceled.

In the case of tag RAM error on valid information write the event is flagged through **ERRWT** to the event monitor. Subsequent reads to that line are Unpredictable.

Clean and Invalidate maintenance operation

In the case of tag parity error or RAM error on cache lookup the event is flagged through **PARRT** or **ERRRT** to the event monitor. The clean operation is canceled and the line is invalidated only for Clean and Invalidate by Index operation. There is no risk of an unexpected hit.

In the case of data parity error or RAM error on data read the event is flagged through **PARRD** or **ERRRD** to the event monitor. The clean operation is cancelled and the line is invalidated only for Clean and Invalidate by Index operation. There is no risk of an unexpected hit.

In the case of tag RAM error on valid information write the event is flagged through **ERRWT** to the event monitor. Subsequent reads to that line are Unpredictable.

Appendix A
Signal Descriptions

This appendix describes the cache controller signals. It contains the following section:

- *Cache controller signals* on page A-2.

A.1 Cache controller signals

This section describes the cache controller signals:

- *Slave port 0 signals*
- *Master port 0 signals* on page A-3
- *Slave port 1 signals* on page A-5
- *Master port 1 signals* on page A-6
- *Slave port 2 signals* on page A-8
- *Master port 2 signals* on page A-9
- *Data RAM interface signals* on page A-11
- *Tag RAM interface signals* on page A-12
- *Dirty RAM interface signals* on page A-13
- *Cache controller event bus* on page A-13
- *Cache controller MBIST interface* on page A-15
- *Miscellaneous signals* on page A-16.

A.1.1 Slave port 0 signals

Table A-1 shows the slave port 0 signals.This assumes SO is present after synthesis.

Table A-1 Slave port 0 signals

Signals	Input/ output	Description
CLKENS0	Input	**CLK** enable for slave port 0
HADDRS0[31:0]	Input	Address bus
HBSTRBS0[7:0]	Input	Byte lane strobes
HBURSTS0[2:0]	Input	Burst length: 000 Single 001 Incr 010 Wrap4 011 Incr4 100 Wrap 8 101 Incr 8 110 Wrap 16 111 Incr 16
HMASTLOCKS0	Input	Locked transfer request

Table A-1 Slave port 0 signals (continued)

Signals	Input/ output	Description
HMASTERS0[3:0]	Input	Master ID for requested access
HPROTS0[5:0]	Input	Protection information associated with transfer
HSIDEBANDS0[3:0]	Input	Signals L1 and sharable attributes
HSIZES0[2:0]	Input	Size of the AHB transfer: Bit [2] is ignored. Bits[1:0] 00 8-bit 01 16-bit 10 32-bit 11 64-bit
HTRANSS0[1:0]	Input	AHB transfer type: 00 IDLE 01 BUSY 10 NSEQ 11 SEQ
HUNALIGNS0	Input	Unaligned access signal
HRDATAS0[63:0]	Output	Read data bus
HREADYS0	Output	Driven LOW to extend transfer
HRESPS0[2:0]	Output	AHB transfer response

A.1.2 Master port 0 signals

Table A-2 shows the master port 0 signals. This assumes MO is present after synthesis

Table A-2 Master port 0 signals

Signals	Input/ output	Description
HCLKENM0	Input	**HCLK** enable for master port 0
HRDATAM0[63:0]	Input	Read data bus
HREADYM0	Input	Ready signal

Table A-2 Master port 0 signals (continued)

Signals	Input/ output	Description
HRESPM0[2:0]	Input	AHB transfer response
HADDRM0[31:0]	Output	Address bus
HBSTRBM0[7:0]	Output	Byte lane strobes
HBURSTM0[2:0]	Output	Burst length 000 Single 001 Incr 010 Wrap4 011 Incr4 100 Wrap 8 101 Incr 8 110 Wrap 16 111 Incr 16
HMASTERM0[3:0]	Output	Master ID for requested access **HMASTERM0** information is only valid for L2 noncacheable nonbufferable accesses and linefills (read miss). For buffered writes, line evictions, and linefills caused by write misses (write allocate region).
HMASTLOCKM0	Output	Locked transfer request
HPROTM0[5:0]	Output	Protection information associated with a transfer
HSIDEBANDM0[3:0]	Output	Signals L1 and sharable attributes
HSIZEM0[2:0]	Output	Size of the AHB transfer
HTRANSM0[1:0]	Output	AHB transfer type: 00 IDLE 01 BUSY 10 NSEQ 11 SEQ
HUNALIGNM0	Output	Unaligned access signal

A.1.3 Slave port 1 signals

Table A-3 shows the slave port 1 signals.

Table A-3 Slave port 1 signals

Signals	Input/ output	Description
CLKENS1	Input	**CLK** enable for slave port 1
HADDRS1[31:0]	Input	Address bus
HBSTRBS1[7:0]	Input	Byte lane strobes
HBURSTS1[2:0]	Input	Burst length 000 Single 001 Incr 010 Wrap4 011 Incr4 100 Wrap 8 101 Incr 8 110 Wrap 16 111 Incr 16
HMASTERS1[3:0]	Input	Master ID for requested access
HMASTLOCKS1	Input	Locked transfer request
HPROTS1[5:0]	Input	Protection information associated with transfer
HSELRS1	Input	Chip select for accessing internal registers
HSIDEBANDS1[3:0]	Input	Signals L1 and sharable attributes
HSIZES1[2:0]	Input	Size of the AHB transfer Bit [2] is ignored. Bits[1:0] 00 8-bit 01 16-bit 10 32-bit 11 64-bit

Table A-3 Slave port 1 signals (continued)

Signals	Input/ output	Description
HTRANSS1[1:0]	Input	AHB transfer type: 00 IDLE 01 BUSY 10 NSEQ 11 SEQ
HUNALIGNS1	Input	Unaligned access signal
HWDATAS1[63:0]	Input	Write data bus
HWRITES1	Input	Write signal
HRDATAS1[63:0]	Output	Read data bus
HREADYS1	Output	Driven LOW to extend transfer
HRESPS1[2:0]	Output	AHB transfer response

A.1.4 Master port 1 signals

Table A-4 shows the master port 1 signals.

Table A-4 Master port 1 signals

Signals	Input/ output	Description
HCLKENM1	Input	**HCLK** enable for Master port 1
HRDATAM1[63:0]	Input	Read data bus
HREADYM1	Input	Ready signal
HRESPM1[2:0]	Input	AHB transfer response
HADDRM1[31:0]	Output	Address bus
HBSTRBM1[7:0]	Output	Byte lane strobes

Table A-4 Master port 1 signals (continued)

Signals	Input/ output	Description
HBURSTM1[2:0]	Output	Burst length 000 Single 001 Incr 010 Wrap4 011 Incr4 100 Wrap 8 101 Incr 8 110 Wrap 16 111 Incr 16
HMASTLOCKM1	Output	Locked transfer request
HMASTERM1[3:0]	Output	Master ID for requested access **HMASTERM1** information is only valid for L2 noncacheable nonbufferable accesses and linefills (read miss). **HMASTERM1** takes `0xF` as its default value for buffered writes, line evictions, linefills caused by write misses and the write allocate region.
HPROTM1[5:0]	Output	Protection information associated with a transfer. **HPROTM1** information is always valid except for line evictions and linefills caused by a write miss, write allocate region. **HPROTM1** takes `0x0F` as its default value in these cases.
HSIDEBANDM1[3:0]	Output	Signals L1 and sharable attributes. **HSIDEBANDM1** information is always valid except for line evictions and linefills caused by write miss, the write allocate region. **HSIDEBANDM1** takes `0xE` as its default value in these cases.
HSIZEM1[2:0]	Output	Size of the AHB transfer Bit [2] is ignored. Bits[1:0] 00 8-bit 01 16-bit 10 32-bit 11 64-bit

Table A-4 Master port 1 signals (continued)

Signals	Input/ output	Description
HTRANSM1[1:0]	Output	AHB transfer type: 00 IDLE 01 BUSY 10 NSEQ 11 SEQ
HUNALIGNM1	Output	Unaligned access signal
HWDATAM1[63:0]	Output	Write data bus
HWRITEM1	Output	Write signal

A.1.5 Slave port 2 signals

Table A-5 shows the slave port 2 signals. This assumes S2 is present after synthesis

Table A-5 Slave port 2 signals

Signals	Input/ output	Description
CLKENS2	Input	**CLK** enable for Slave port 2
HADDRS2[31:0]	Input	Address bus
HBSTRBS2[7:0]	Input	Byte lane strobes
HBURSTS2[2:0]	Input	Burst length 000 Single 001 Incr 010 Wrap4 011 Incr4 100 Wrap 8 101 Incr 8 110 Wrap 16 111 Incr 16
HMASTERS2[3:0]	Input	Master ID for requested access
HMASTLOCKS2	Input	Locked transfer request
HPROTS2[5:0]	Input	Protection information associated with transfer

Table A-5 Slave port 2 signals (continued)

Signals	Input/ output	Description
HSELRS2	Input	Chip select for accessing internal registers
HSIDEBANDS2[3:0]	Input	Signals L1 and sharable attributes
HSIZES2[2:0]	Input	Size of the AHB transfer Bit [2] is ignored. Bits[1:0] 00 8-bit 01 16-bit 10 32-bit 11 64-bit
HTRANSS2[1:0]	Input	AHB transfer type: 00 IDLE 01 BUSY 10 NSEQ 11 SEQ
HUNALIGNS2	Input	Unaligned access signal
HWDATAS2[63:0]	Input	Write data bus
HREADYS2	Output	Driven LOW to extend transfer
HRESPS2[2:0]	Output	AHB transfer response

A.1.6 Master port 2 signals

Table A-6 shows the master port 2 signals. This assumes M2 is present after synthesis

Table A-6 Master port 2 signals

Signals	Input/ output	Description
HCLKENM2	Input	**HCLK** enable for Master port 2
HREADYM2	Input	Ready signal
HRESPM2[2:0]	Input	AHB transfer response
HADDRM2[31:0]	Output	Address bus

Table A-6 Master port 2 signals (continued)

Signals	Input/ output	Description
HBSTRBM2[7:0]	Output	Byte lane strobes
HBURSTM2[2:0]	Output	Burst length 000 Single 001 Incr 010 Wrap4 011 Incr4 100 Wrap 8 101 Incr 8 110 Wrap 16 111 Incr 16
HMASTLOCKM2	Output	Locked transfer request
HMASTERM2[3:0]	Output	Master ID for requested access **HMASTERM2** information is only valid for L2 noncacheable nonbufferable accesses and linefills (read miss). **HMASTERM2** information is always valid except for buffered write and line evictions. In these two cases **HMASTERM2** takes `0xF` as its default value.
HPROTM2[5:0]	Output	Protection information associated with a transfer. **HPROTM2** information is always valid except for line evictions. **HPROTM2** takes `0x0F` as its default value in this case.
HSIDEBANDM2[3:0]	Output	Signals L1 and sharable attributes. **HSIDEBANDM2** information is always valid except for line evictions. **HSIDEBANDM2** takes `0xE` as its default value in this case.
HSIZEM2[2:0]	Output	Size of the AHB transfer Bit [2] is ignored. Bits[1:0] 00 8-bit 01 16-bit 10 32-bit 11 64-bit

Table A-6 Master port 2 signals (continued)

Signals	Input/ output	Description
HTRANSM2[1:0]	Output	AHB transfer type: 00 IDLE 01 BUSY 10 NSEQ 11 SEQ
HUNALIGNM2	Output	Unaligned access signal
HWDATAM2[63:0]	Output	Write data bus

A.1.7 Data RAM interface signals

Table A-7 shows the cache controller data RAM interface signals.

Table A-7 Data RAM interface signals

Signals	Input/ output	Description
DATAERR	Input	Data RAM error
DATAPRD[31:0]	Input	Data RAM parity read data
DATARD[255:0]	Input	Data RAM read data
DATAADDR[15:0]	Output	Data RAM address
DATACS	Output	Data RAM chip select
DATAEN[31:0]	Output	Data RAM byte write enables
DATAnRW	Output	Current access to data RAM is a write
DATAPEN[31:0]	Output	Data parity RAM byte write enables
DATAPnRW	Output	Current access to data parity RAM is a write
DATAPWD[31:0]	Output	Data RAM parity write data
DATAWD[255:0]	Output	Data RAM write data

A.1.8 Tag RAM interface signals

Table A-8 shows the tag RAM interface signals.

Table A-8 Tag RAM interface signals

Signal name	Input/ output	Description
TAGERR[7:0]	Input	Tag RAM error
TAGPRD[7:0]	Input	Tag RAM parity read data
TAGRD0[18:0]	Input	Tag RAM 0 read data
TAGRD1[18:0]	Input	Tag RAM 1 read data
TAGRD2[18:0]	Input	Tag RAM 2 read data
TAGRD3[18:0]	Input	Tag RAM 3 read data
TAGRD4[18:0]	Input	Tag RAM 4 read data
TAGRD5[18:0]	Input	Tag RAM 5 read data
TAGRD6[18:0]	Input	Tag RAM 6 read data
TAGRD7[18:0]	Input	Tag RAM 7 read data
TAGADDR[12:0]	Output	Tag RAM address
TAGCS[7:0]	Output	Tag RAM chip selects
TAGEN	Output	Tag RAM write enable This is a single-bit signal that applies to all eight tag RAMs
TAGPWD	Output	Tag RAM parity write data
TAGWD[18:0]	Output	Tag RAM write data

A.1.9 Dirty RAM interface signals

Table A-9 shows the dirty RAM interface signals.

Table A-9 Dirty RAM interface signals

Signal name	Input/ output	Description
DIRTYRD[15:0]	Input	Read data from dirty RAM
DIRTYCS	Output	Chip select for dirty RAM
DIRTYEN[15:0]	Output	Write enables for dirty RAM
DIRTYnRW	Output	Current access to dirty RAM is a write
DIRTYWD[15:0]	Output	Write data for dirty RAM

A.1.10 Cache controller event bus

Table A-10 shows the cache controller event bus output signals.

Table A-10 Cache controller event bus output signals

Signal name	Notes
BWABT	Buffered write abort or abort caused by write-allocate buffer read
CO	Cache line or half line eviction
DRHIT	Data read hit in the L2 cache controller[a]
DRREQ	Data read look-up to the L2 cache controller. This can subsequently hit or miss[a].
DWHIT	Data write buffer write hit in the L2 cache
DWREQ	Data write buffer write look-up to the L2 cache with Write-Through attribute. This can subsequently hit or miss.
DWTREQ	Data write buffer write look-up with Write-Through attribute
ERRRD	Error on L2 data RAM read
ERRRT	Error on L2 tag RAM read
ERRWD	Error on L2 data RAM write
ERRWT	Error on L2 tag RAM write

Table A-10 Cache controller event bus output signals (continued)

Signal name	Notes
IRHIT	Instruction read hit in the L2 cache controller[a]
IRREQ	Instruction read look-up to the L2 cache. This can subsequently hit or miss.
PARRD	Parity error on L2 data RAM read
PARRT	Parity error on L2 tag RAM read
WA	Allocation into the L2 cache caused by a write, with Write-Allocate attribute, miss

a. When data read requests hit in the slave line read buffer, no events are generated through these pins.

A.1.11 Cache controller MBIST interface

Table A-11 shows the cache controller MBIST inputs and outputs.

Table A-11 Cache controller MBIST block inputs and outputs

Signal	Input/ output	Notes
MBISTDOUT[63:0]	Input	MBIST data out, to cache controller **MBISTDOUT[63:0]** = MBIST data out for data RAM **MBISTDOUT[31:0]** = MBIST data out for data parity RAM **MBISTDOUT[19:0]** = MBIST data out for tag RAM **MBISTDOUT[15:0]** = MBIST data out for dirty RAM
MBISTADDR[17:0]	Output	MBIST address **MBISTADDR[17:0]** used for data RAM, two LSBs used as doubleword select **MBISTADDR[17:2]** used for data parity RAM **MBISTADDR[14:2]** used for Tag and Dirty RAMs
MBISTCE[10:0]	Output	MBIST RAM Chip Enables, for writes **MBISTCE[0]** = data RAM chip enable **MBISTCE[1]** = tag RAM 0 chip enable **MBISTCE[2]** = tag RAM 1 chip enable **MBISTCE[3]** = tag RAM 2 chip enable **MBISTCE[4]** = tag RAM 3 chip enable **MBISTCE[5]** = tag RAM 4 chip enable **MBISTCE[6]** = tag RAM 5 chip enable **MBISTCE[7]** = tag RAM 6 chip enable **MBISTCE[8]** = tag RAM 7 chip enable **MBISTCE[9]** = dirty RAM chip enable **MBISTCE[10]** = data parity RAM chip enable

Table A-11 Cache controller MBIST block inputs and outputs (continued)

Signal	Input/ output	Notes
MBISTDCTL[12:0]	Output	MBIST Control, for reads **MBISTDCTL[1:0]** = MBIST data select for 64 bits of 256-bit wide data RAM **MBISTDCTL[2]** = MBIST RAM select for Data RAM **MBISTDCTL[3]** = MBIST RAM select for tag RAM 0 **MBISTDCTL[4]** = MBIST RAM select for tag RAM 1 **MBISTDCTL[5]** = MBIST RAM select for tag RAM 2 **MBISTDCTL[6]** = MBIST RAM select for tag RAM 3 **MBISTDCTL[7]** = MBIST RAM select for tag RAM 4 **MBISTDCTL[8]** = MBIST RAM select for tag RAM 5 **MBISTDCTL[9]** = MBIST RAM select for tag RAM 6 **MBISTDCTL[10]** = MBIST RAM select for tag RAM 7 **MBISTDCTL[11]** = MBIST RAM select for dirty RAM **MBISTDCTL[12]** = MBIST RAM select for data parity RAM
MBISTDIN[63:0]	Output	MBIST data in, to cache controller **MBISTDIN[63:0]** = MBIST data in for data RAM **MBISTDIN[31:0]** = MBIST data in for data parity RAM **MBISTDIN[19:0]** = MBIST data in for tag RAM **MBISTDIN[15:0]** = MBIST data in for dirty RAM
MBISTWE	Output	MBIST Write Enable
MTESTON	Output	MBIST Mode Enable

A.1.12 Miscellaneous signals

Table A-12 shows miscellaneous signals.

Table A-12 Miscellaneous signals

Signal name	Input/ output	Description
BIGEND	Input	CPU core endianness, BE-32
CACHEID[5:0]	Input	Value read on CACHE ID Register 0 bits [15:10] Reserved for layout Implementor. See *ID Register* on page 2-7 for details.
CLK	Input	Main CPU or Cache controller clock

Table A-12 Miscellaneous signals (continued)

Signal name	Input/ output	Description
HCLK	Input	Main memory clock, the same as **CLK** for synchronous mode
HSELSR2EN	Input	Enable writes to internal registers through S2 port
HSYNCEN[a]	Input	Synchronous mode enable 0 = The cache controller is in asynchronous mode of operation 1 = The cache controller is in synchronous mode of operation
MASTNUM[1:0]	Input	Number of master ports: 00: one master, M101: two masters, M0 and M1 1x: three masters, M0, M1, and M2
SLAVENUM[1:0]	Input	Number of slave ports: 00: one slave, S101: two slaves, S0 and S11x: three slaves, S0, S1, and S2
nHRESET	Input	Reset signal for **HCLK** clock domain logic, active LOW
nRESET	Input	Reset signal for **CLK** clock domain logic, active LOW
SIZEM0	Input	Data bus size: 0 = 32 bits 1 = 64 bits
SIZEM1	Input	Data bus size: 0 = 32 bits 1 = 64 bits
SIZEM2	Input	Data bus size: 0 = 32 bits 1 = 64 bits
SIZES0	Input	Data bus size: 0 = 32 bits 1 = 64 bits
SIZES1	Input	Data bus size: 0 = 32 bits 1 = 64 bits

Table A-12 Miscellaneous signals (continued)

Signal name	Input/ output	Description
SIZES2	Input	Data bus size: 0 = 32 bits 1 = 64 bits
TESTEN	Input	Test mode enable Used to enable clocks of unused slaves or masters for test
CLKOUT	Output	Output clock signal for RAMs
DIRTYLAT[2:0]	Output	Value set in bits[11:9] of the Auxiliary Control Register[b]
IDLE	Output	The cache controller is idle
RDATALAT[2:0]	Output	Value set in bits[2:0] of the Auxiliary Control Register[c]
TAGLAT[2:0]	Output	Value set in bits[8:6] of the Auxiliary Control Register[d]
WDATALAT[2:0]	Output	Value set in bits[5:3] of the Auxiliary Control Register[e]

a. **HSYNCEN** does not exist if the asynchronous interface has been removed by synthesis.
b. Described in *Auxiliary Control Register* on page 2-9.
c. Described in *Auxiliary Control Register* on page 2-9.
d. Described in *Auxiliary Control Register* on page 2-9.
e. Described in *Auxiliary Control Register* on page 2-9.

Appendix B
AC Parameters

This appendix describes the cache controller AC parameters. It contains the following section:

- *Cache controller interface signal timing parameters* on page B-2.

B.1 Cache controller interface signal timing parameters

Signal timing parameters are given in:

- *Registered signals*
- *Unregistered signals*.

B.1.1 Registered signals

To ensure ease of integration of the cache controller into embedded applications, and to simplify synthesis flow, the following design techniques have been used:

- a single rising edge clock times all activity
- all signals and buses are unidirectional
- all inputs are required to be synchronous to the relevant clock, **CLK**, or **HCLK**.

These techniques simplify the definition of the top-level cache controller signals because all outputs change from the rising edge and all inputs are sampled with the rising edge of the clock. In addition, all signals are either input or output only. Bidirectional signals are not used.

B.1.2 Unregistered signals

The unregistered input signals are:

- **nRESET,** and **nHRESET**
- **CLKENS0**, **CLKENS1**, and **CLKENS2**
- **HCLKENM0**, **HCLKENM1**, and **HCLKENM2**
- **HREADYM0**, **HREADYM1**, **HREADYM2**
- **DATARD, DATAPRD, and DATAERR**.

The unregistered output signals are:

- **HREADYS0**, **HREADYS1**, and **HREADYS2**
- **HRESPS0**, **HRESPS1**, and **HRESPS2**.

Figure B-1 on page B-3 shows the cache controller target timing parameters for unregistered signals. The timing parameter T is the internal clock latency of the clock buffer tree, dependent on process technology and design parameters. Timing parameters ending with suffix h represent hold times. Timing parameters ending with suffix d represent delay times. Contact your silicon supplier for more details.

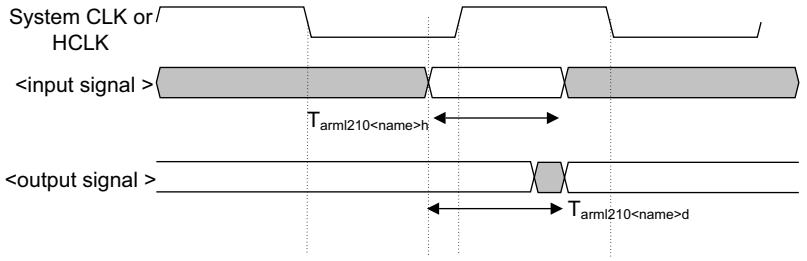


Figure B-1 Target timing parameters for unregistered signals

Note

Actual clock frequencies and input and output timing constraints vary according to application requirements and the silicon process technologies used. The maximum operating clock frequencies attained by ARM devices increases over time as a result.

Appendix C
Event Monitor

This appendix describes the cache controller event monitor signals. It contains the following sections:

- *Cache controller event monitor* on page C-2
- *Programmer's model* on page C-4
- *Implementation details* on page C-9
- *Event monitor signals* on page C-12.

C.1 Cache controller event monitor

The cache controller is delivered with an event monitor block template that is intended to be plugged onto the cache controller event bus. It can be reshaped by the licensee to meet their system requirements.

The addition of an event monitor block in a system that contains an ARM processor and a cache controller enables monitoring of events and errors in the RAMs that make up the cache controller. This information can be used to tune the overall system performance.

To optimize the performance of a system, the cache controller event monitor block provides four event counters, EMN0-EMN3. They can be used to count the instances of four different events selected from a list of possible external events, through the cache controller event bus, or internal events such as overflows or clock edges. Each counter is a 32-bit counter and has its own interrupt generation logic. By combining different statistics, you can obtain a variety of performance metrics.

The example system described in this Appendix has the following features:

- one 32-bit monitor control register, EMMC
- one 32-bit counter status register, EMCS
- four 32-bit counter configuration registers, EMCC0-EMCC3
- four 32-bit read-only event counters, EMC0-EMC3
- AHB-Lite interface, for internal register accesses
- ability to track the following events
  - CPU instruction read request to L210 cache
  - CPU instruction read hit in L210 cache
  - CPU data read request to L210 cache
  - CPU data read hit in L210 cache
  - CPU data write request to L210 cache, not write-through
  - CPU data write request to L210 cache, write-through
  - CPU data write hit in L210 cache
  - L210 buffered write abort
  - L210 cache half-line eviction, two events for one full-line eviction
  - allocation to the L210 caused by write transaction, write-allocate
  - error on data RAM read access

  - — error on data RAM write access
  - — error on tag RAM read access
  - — error on tag RAM write access
  - — parity error on data RAM read access
  - — parity error on tag RAM read transaction.
- intended to run at same speed as L210 block, 300Mhz target in 0.18µm process.
- synchronous or asynchronous AHB-Lite interface.

C.2 Programmer's model

The cache controller event monitor is accessed through a set of memory-mapped registers that occupy a relocatable 4KB region of memory. The base address of the cache controller is not fixed, but is determined by an AHB decoder on the slave buses and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.

The registers are accessed through an AHB-Lite slave interface. The cache controller event monitor contains ten registers. Table C-1 shows this.

Table C-1 Event monitor registers

Register	Function	Address
EMMC	Event Monitor Control	BASE + 0x000
EMCS	Counter status register	BASE + 0x004
EMCC0	Counter 0 configuration	BASE + 0x100
EMCC1	Counter 1 configuration	BASE + 0x104
EMCC2	Counter 2 configuration	BASE + 0x108
EMCC3	Counter 3 configuration	BASE + 0x10C
EMC0	(read-only) Counter 0	BASE + 0x200
EMC1	(read-only) Counter 1	BASE + 0x204
EMC2	(read-only) Counter 2	BASE + 0x208
EMC3	(read-only) Counter 3	BASE + 0x20C

C.2.1 Event Monitor Control Register, EMMC

The control register for the cache controller event monitor should be accessed using a read-modify-write sequence. The EMMC is used to:

- enable the event monitor block
- control interrupt generation
- reset counters to zero.

Table C-2 shows the bit fields.

Table C-2 EMMC bit fields descriptions

Bit	Field	Description
[31:12]	Reserved SBZ/ RAZ	-
[11:8]	Counter Reset Always Read as zero	Bit11: EMC3 reset. Corresponding registers are cleared when bit is writtenBit 10: EMC2 reset to 1Bit 9: EMC1 reset to 1Bit 8: EMC0 reset to 1
[7:6]	Reserved SBZ/ RAZ	-
[5:3]	Edge-sensitive interrupt pulse duration	b000:1 **CLK** cycle b001:2 **CLK** cycles b010:4 **CLK** cycles . . . b111:128 **CLK** cycles
[2]	Interrupt polarity	0: Interrupt signal is active LOW, default1: Interrupt signal is active HIGH
[1]	Interrupt Type	0: Level sensitive interrupt line remains active until all Counter Flags are cleared, default 1: Edge sensitive, interrupt line active for n **CLK** cycles. n is defined by bits[5:3]
[0]	Event monitor enable	Event monitor enable 0: Event Monitor Disabled, default 1: Event Monitor enabled

C.2.2 Counter Status Register, EMCS

The Counter Status Register contains a set of flags that indicate if the corresponding counter flag set condition, as described in Table C-3 on page C-6, has occurred. It is used to determine which counter or counters cause an interrupt if interrupt generation is enabled.

If the interrupt is programmed to be level sensitive, the interrupt line remains active until all flags in the Counter Status Register are cleared. Table C-3 describes the Counter Status Register bits.

Table C-3 Counter Status Register

Bit	Field	Description
[31:4]	Reserved	*Should be Zero* (SBZ)/RAZ
[3]	EMC3 Flag 0	0 = Counter Flag set condition has not occurred 1 = Counter Flag set condition has occurred Counter Flags are cleared when written to 1
[2]	EMC2 Flag 1	
[1]	EMC1 Flag	
[0]	EMC0 Flag	

C.2.3 Counter Configuration Registers, EMCCx

For each counter, a configuration register exists so that user can define:

- counter event source
- counter Flag set condition, flag is in Counter Status Register
- interrupt enable.

Table C-4 shows the Counter Configuration Registers.

Table C-4 Counter Configuration Register

Bit	Field	Description
[31:4]	Reserved	SBZ/RAZ
[6:2]	Counter event source	Table C-5 on page C-7 gives the encodings.
[1]	Counter Flag set condition	0 = Counter flag set on overflow, default 1 = Counter flag set on increment
[0]	Counter interrupt generation enable	0 = No interruption generated by the counter, default 1= An interrupt is generated when counter flag set condition is met

C.2.4 Event sources encodings

Table C-5 shows how all event sources for event monitor counters are encoded.

Table C-5 Event sources encodings

Event encoding	L210 Pin	Event description
b00000	N/A	Counter Disabled
b00001	BWABT	Buffered write abort
b00010	CO	L210 half-line eviction, two events for one full-line eviction
b00011	DRHIT	Data read hit
b00100	DRREQ	Data read request
b00101	DWHIT	Data write hit
b00110	DWREQ	Data write request (not Write-Through)
b00111	DWTREQ	Data write request (Write-Through)
b01000	ERRRD	Error on L210 cache data RAM read
b01001	ERRRT	Error on L210 cache tag RAM read
b01010	ERRWD	Error on L210 cache data RAM write
b01011	ERRWT	Error on L210 cache tag RAM write
b01100	IRHIT	Instruction read hit
b01101	IRREQ	Instruction read request
b01110	PARRD	Parity error on L210 cache data RAM read
b01111	PARRT	Parity error on L210 cache tag RAM read
b10000	WA	Write allocate (L210 cache allocation caused to write transaction)
b10001	ERRRD ERRRT ERRWD ERRWT	Any RAM error (OR of ERRRD)

Table C-5 Event sources encodings (continued)

Event encoding	L210 Pin	Event description
b10010	PARRD PARRT Any parity error (OR of PARRD	PARRT
b10011	N/a	EMC3 overflow
b10100	N/a	EMC2 overflow
b10110	N/a	EMC0 overflow
b10111	N/a	**CLK** cycle (counter incremented on every **CLK** cycle)
b11000 -b11111	N/a	Counter disabled

C.3 Implementation details

The cache controller event monitor block appears as an AHB slave to the CPU. All transactions to the cache controller event monitor are seen as single 32-bit AHB transactions. HSIZE, HBURST, and HPROT pins are not required.

The slave always answers with **HRESP** = 00 (OKAY). This means that:

- Non 32-bit transactions, **HSIZE** b010, is always treated as a 32-bit transaction.
- A read to an unmapped register always returns b0000.
- A write to an unmapped register is silently discarded.

C.3.1 Event monitor clocking

The cache controller and the cache controller event monitor block must be clocked with the same clock. It is intended that the cache controller and the event monitor are clocked with the same clock as the CPU core, but an asynchronous interface, as a synthesis option, enables core and event monitor to have different clocks. See *Asynchronous interface* on page 4-4 for a description of the interface options.

C.3.2 Interrupts

The user can configure the cache controller event monitor to control precisely how interrupts are generated. Interrupt generation set-up is made in four steps

1. Interrupt signal set-up.

   The interrupt line (INT) can be configured to be level-sensitive, interrupt tied to active level until interrupt is cleared, or edge sensitive, one pulse to active level, lasting a programmable number of **CLK** cycles and the active level is also configurable. In that way, the cache controller event monitor can be directly connected to the CPU, level-sensitive active LOW interrupt, or to any kind of interrupt controller.

2. Counter set-up.

   For each counter, an incrementing trigger is chosen between all possible events.

3. Counter Flag set condition set-up.

   Counter flags can be set on two conditions, increment or overflow. CCNT Flag is only set on overflow.

4. Interruption enable.

   Set counter flags do not generate an interrupt until their interrupt enable bit is set.

Priority is given to the event. If a flag set condition occurs at the same cycle as the related flag is cleared by the CPU, the flag remains set. In this way a level-sensitive interrupt remains active.

Edge-sensitive interrupts obey these rules:

- An interrupt pulse is started each time a flag is set, even if the flag, or the others, was already set.
- If a flag is set during a pending interrupt pulse, the pulse is not strengthened.

C.3.3 AHB clocking

The AHB interface has its own clock, **HCLK**, that enables the memory system to run at a different frequency to the CPU core, the cache controller, and the event monitor. With respect to **CLK**, **HCLK** can be either:

Synchronous

HCLK frequency is the same or a submultiple of **CLK** frequency and the skew between the two clocks is minimal. **HSYNCEN** =1 in this case.

Asynchronous

There is no phase or frequency relation between **HCLK** and **CLK**. Like the ARM1136 core and the cache controller, the event monitor contains optional synchronization logic, synthesis option with **HSYNCEN** input pin set to 0, that enables asynchronous behavior.

In addition to **HCLK**, **HCLKEN** input is used to reduce the transfer rate on AHB bus to a ratio of **HCLK**. Figure C-1 shows this. All signals on AHB bus are considered valid, and so can be sampled, on **HCLK** rising edge when **HCLKEN** is HIGH. No assumption is made on synchronicity between **CLK** and **HCLK** in that case.

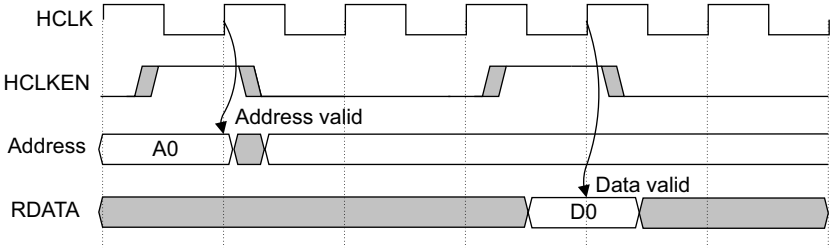


Figure C-1 HCLKEN usage for a read

Synchronous interface

When **HSYNCEN** is set to HIGH, `l2cc_em module` is fully synchronous. **CLK** and **HCLK** should be connected together to the same clock.

The AHB slave interface always responds with one wait state.

Asynchronous interface

When **HSYNCEN** is set to LOW, `l2cc_em module` is operating in asynchronous mode. No phase relation is assumed between **HCLK** and **CLK**.

The AHB slave interface generates wait states, by asserting **HREADY** LOW, as long as needed to ensure that the request is seen by the register module and the acknowledge is forwarded back to the AHB interface.

C.3.4 AHB-Lite slave port interface signals

The cache controller event monitor block appears as an AHB slave to the CPU. All transactions to the cache controller event monitor are seen as single 32-bit AHB transactions. Table C-6 shows the AHB-Lite slave port interface signals. **HSIZE**, **HBURST**, and **HPROT** pins are not required.

Table C-6 AHB-Lite slave port interface signals

Signal	Input/output	Notes
HADDR[11:0]	Input	Address bus
HSEL	Input	Slave select
HTRANS1	Input	AHB transfer type Connect to bit 1 of the HTRANS bus on the AHB.
HWDATA[31:0]	Input	Write data bus
HWRITE	Input	Read/write signal
HRDATA[31:0]	Output	Read data bus
HREADYin	Input	Common **HREADY** signal
HREADYout	Output	Driven LOW to extend transfer
HRESP[1:0]	Output	AHB transfer response

C.4 Event monitor signals

Table C-7 shows the inputs to the event monitor from cache controller.

Table C-7 Event monitor input signals

Signal name	Notes
BWABT	Buffered write abort or abort caused by write-allocate buffer read
CO	L210 line eviction
DRHIT	Data read hit
DRREQ	Data read request
DWHIT	Data write hit
DWREQ	Data write request
DWTREQ	Data write request with Write-Through attribute
ERRRD	Error on L210 cache data RAM read
ERRRT	Error on L210 cache tag RAM read
ERRWD	Error on L210 cache data RAM write
ERRWT	Error on L210 cache tag RAM write
IRHIT	Instruction read hit
IRREQ	Instruction read request
PARRD	Parity error on L210 cache data RAM read
PARRT	Parity error on L210 cache tag RAM read
WA	Write allocate, write caused a linefill to the L210

Appendix D
Master and Slave Port Configurations

This appendix describes the cache controller master and slave port configurations that you can use with an ARM1136 core, and with an ARM1026EJ-S core or ARM926EJ-S core. It also describes the mapping of the salve to master port bursts. It contains the following sections:

- *ARM1136 memory system configurations* on page D-2
- *ARM926EJ-S and ARM1026EJ-S memory systems* on page D-4
- *Mapping of slave to master port bursts* on page D-6.

D.1 ARM1136 memory system configurations

Figure D-1 shows an ARM 1136 memory system using all three slave ports and the signals output on a three-master configuration, a two-master configuration and a single-master configuration.

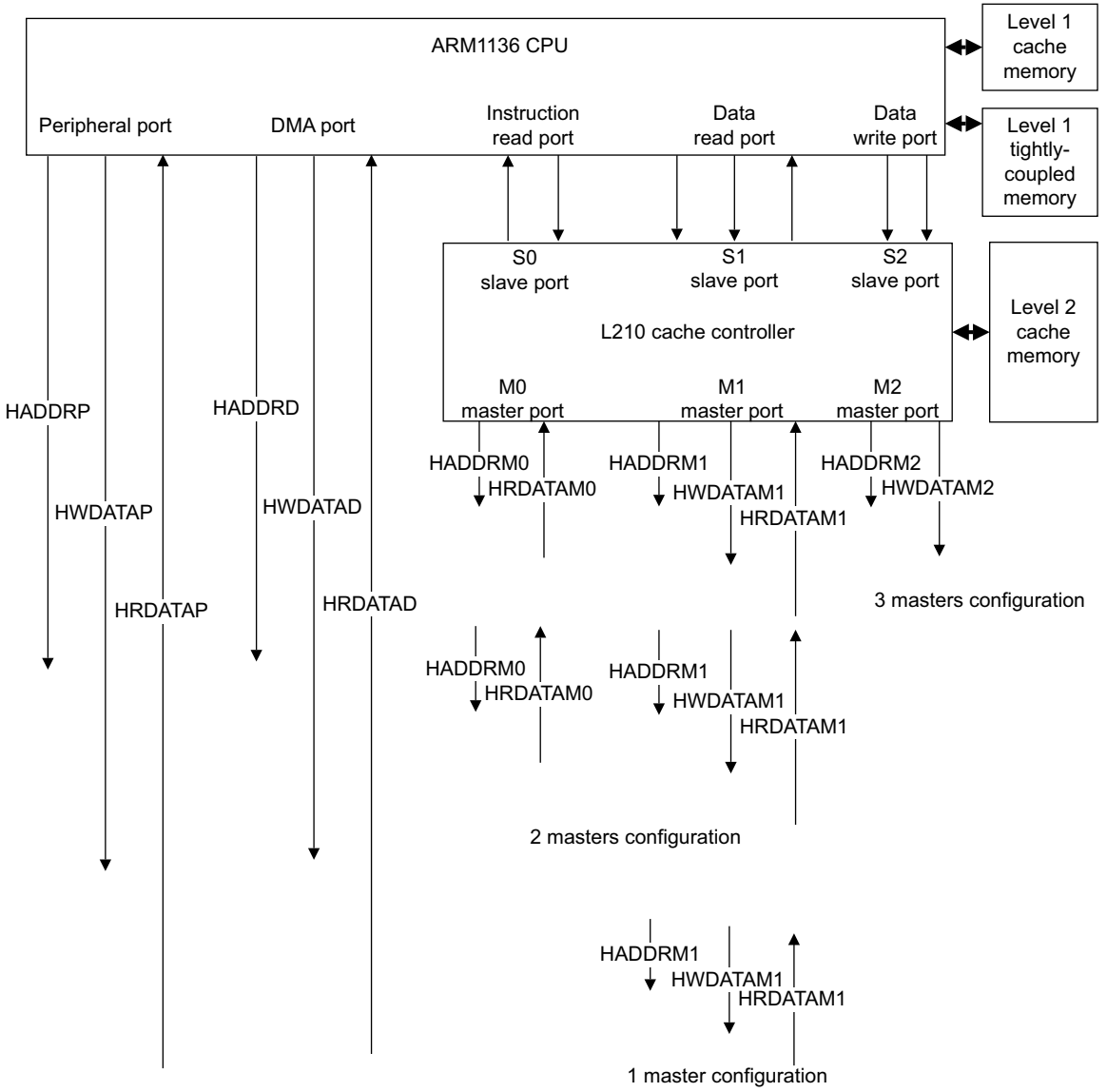


Figure D-1 ARM1136 memory system with cache controller master port configurations

See *Behavior for ARMv6 memory systems* on page 4-14 for information on ARMv6 memory system cache transactions.

D.2 ARM926EJ-S and ARM1026EJ-S memory systems

Figure D-2 on page D-5 shows an ARM1026EJ-S or ARM926EJ-S memory system using 2 slave ports and the signals output on a three-master configuration, a two-master configuration and a single-master configuration. Slave port S2 is unused.

See *Behavior for ARMv5 memory systems* on page 4-10 for information on ARMv5 memory system cache transactions.

Note

A byte-lane converter must be added on the master side for **HUNALIGN/HBSTRB** signal removal in v5 memory systems.

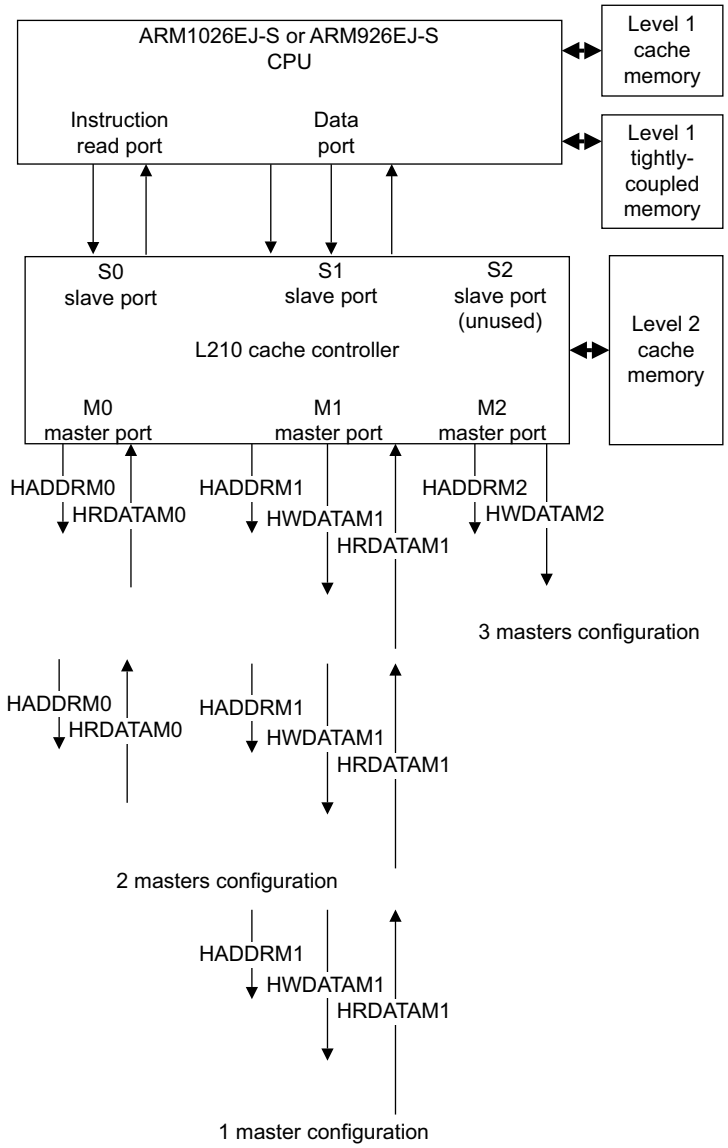


Figure D-2 ARM926EJ-S and ARM1026EJ-S memory system with cache controller master port configurations

D.3 Mapping of slave to master port bursts

This section describes:

- *L2 off or noncacheable read accesses*
- *L2 off or nonbufferable write accesses* on page D-7.

D.3.1 L2 off or noncacheable read accesses

This section describes two states:

- *Wrap access disabled, L2 off or noncacheable read accesses*
- *Wrap access enabled, L2 off or noncacheable read access*.

Wrap access disabled, L2 off or noncacheable read accesses

To achieve wrap access disabled, the Auxiliary Control Register bit[12] is set to 1. When using 64-bit wide master ports, all INCRn burst transactions are kept unchanged. When using 32-bit wide master ports and the size of the burst is a doubleword, then the burst changes are as follows:

- INCR4 => INCR8
- INCR8 => INCR16
- INCR16 => INCR

Table D-1 shows the WRAPn read access conversions on slave ports.

Table D-1 WRAPn read access conversions on slave ports

Size	Master width	WRAP4	WRAP8	WRAP16
Byte	-	INCR4	INCR8	2 x INCR
Halfword	-	INCR4	INCR8	2 x INCR
Word	-	INCR4	INCR8	2 x INCR
Doubleword	64-bit wide master port	INCR4	2 x INCR	2 x INCR
	32-bit wide master port	INCR8	2 x INCR	2 x INCR

Wrap access enabled, L2 off or noncacheable read access

To achieve wrap access enabled, the Auxiliary Control Register bit[12] is set to 0. When using 32-bit wide master ports and the size of the burst is a doubleword, then the burst changes are as follows:

- INCR4 => INCR8

- INCR8 => INCR16
- INCR16 => INCR
- WRAP4 => WRAP8
- WRAP8 => WRAP16
- WRAP16 => 2 x INCR

Note

There are no changes for the other sizes.

D.3.2 L2 off or nonbufferable write accesses

This section describes two states:

- *Wrap access disabled, L2 off or nonbufferable write accesses*
- *Wrap access enabled, L2 off or nonbufferable write accesses*.

Wrap access disabled, L2 off or nonbufferable write accesses

To achieve wrap access disabled, the Auxiliary Control Register bit[12] is set to 1. When using 64-bit wide master ports, all INCRn burst transactions are kept unchanged. When using 32-bit wide master ports and the size of the burst is a doubleword, then the burst changes are as follows:

- INCR4 => INCR8
- INCR8 => INCR16
- INCR16 => INCR

All WRAPn write accesses are converted in two INCR bursts on the master side.

Wrap access enabled, L2 off or nonbufferable write accesses

To achieve wrap access enabled, the Auxiliary Control Register bit[12] is set to 0. When using 32-bit wide master ports and the size of the burst is a doubleword, then the burst changes are as follows:

- INCR4 => INCR8
- INCR8 => INCR16
- INCR16 => INCR
- WRAP4 => WRAP8
- WRAP8 => WRAP16
- WRAP16 => 2 x INCR

Note

There are no changes for the other sizes.

Glossary

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

Abort

A mechanism that indicates to a core that it must halt execution of an attempted illegal memory access. An abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction or data memory. An abort is classified as either a Prefetch Abort, a Data Abort, or an External Abort.

See also Data Abort, External Abort and Prefetch Abort.

Advanced eXtensible Interface (AXI)

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

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

Advanced High-performance Bus (AHB)
A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM Limited recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

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

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

Advanced High-performance Bus (AHB)
The AMBA Advanced High-performance Bus system connects embedded processors such as an ARM core to high-performance peripherals, DMA controllers, on-chip memory, and interfaces. It is a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

AHB-Lite
AHB-Lite is a subset of the full AHB specification. It is intended for use in designs where only a single AHB master is used. This can be a simple single AHB master system or a multi-layer AHB system where there is only one AHB master on a layer.

Aligned
Aligned data items are stored so that their address is divisible by the highest power of two that divides their size. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively. Other related terms are defined similarly.

AMBA
See Advanced Microcontroller Bus Architecture.

Application Specific Integrated Circuit
An integrated circuit that has been designed to perform a specific application function. It can be custom-built or mass-produced.

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

Arithmetic Logic Unit The component of a processor core that performs arithmetic and logic operations.

ARM state A processor that is executing ARM (32-bit) instructions is operating in ARM state.

See also Thumb state.

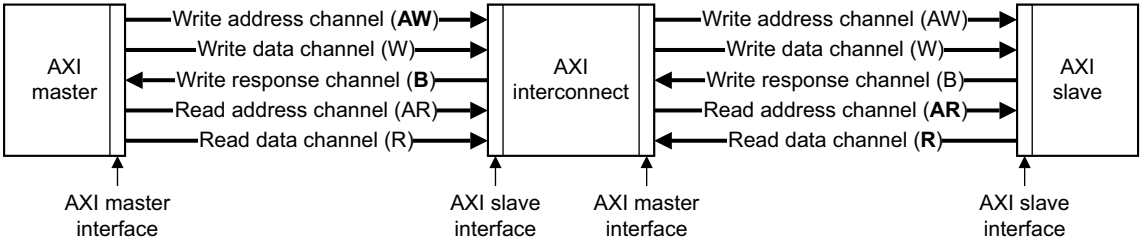
ASIC *See* Application Specific Integrated Circuit.

AXI *See* Advanced eXtensible Interface.

AXI channel order and interfaces

The block diagram shows:

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



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

Active read transaction

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

Active transfer

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

---

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

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

Active write transaction

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

Completed transfer

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

Payload The non-handshake signals in a transfer.

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

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

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

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

Combined issuing capability

The maximum number of active transactions that a master interface can generate. This is specified instead of write or read issuing capability for master interfaces that use a combined storage for active write and read transactions.

Read ID capability

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

Read ID width

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

Read issuing capability

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

Write ID capability

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

Write ID width

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

Write interleave capability

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

Write issuing capability

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

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

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. This is specified instead of write or read acceptance capability for slave interfaces that use a combined storage for active write and read transactions.

Read acceptance capability

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

Read data reordering depth

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

Write acceptance capability

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

Write interleave depth

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

Base register

A register specified by a load or store instruction that is used to hold the base value for the address calculation for the instruction. Depending on the instruction and its addressing mode, an offset can be added to or subtracted from the base register value to form the virtual address that is sent to memory.

Big-endian

Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also Little-endian and Endianness.

Big-endian memory
Memory in which:- a byte or halfword at a word-aligned address is the most significant byte or halfword within the word at that address - a byte at a halfword-aligned address is the most significant byte within the halfword at that address.

See also Little-endian memory.

Block address
An address that comprises a tag, an index, and a word field. The tag bits identify the way that contains the matching cache entry for a cache hit. The index bits identify the set being addressed. The word field contains the word address that can be used to identify specific words, halfwords, or bytes within the cache entry.

See also Cache terminology diagram on the last page of this glossary.

Burst
A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AHB buses are controlled using the **HBURST** signals to specify if transfers are single, four-beat, eight-beat, or 16-beat bursts, and to specify how the addresses are incremented.

Byte
An 8-bit data item.

Byte lane strobe
An AHB signal, **HBSTRB**, that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of **HBSTRB** corresponds to eight bits of the data bus.

Cache
A block of on-chip or off-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used instructions and/or data. This is done to greatly reduce the average speed of memory accesses and so to increase processor performance.

See also Cache terminology diagram on the last page of this glossary.

Cache hit
A memory access that can be processed at high speed because the instruction or data that it addresses is already held in the cache.

Cache line
The basic unit of storage in a cache. It is always a power of two words in size (usually four or 8 words), and is required to be aligned to a suitable memory boundary.

See also Cache terminology diagram on the last page of this glossary.

Cache line index
The number associated with each cache line in a cache way. Within each cache way, the cache lines are numbered from 0 to (set associativity) -1.

See also Cache terminology diagram on the last page of this glossary.

Cache lockdown

To fix a line in cache memory so that it cannot be overwritten. Cache lockdown enables critical instructions and/or data to be loaded into the cache so that the cache lines containing them will not subsequently be reallocated. This ensures that all subsequent accesses to the instructions/data concerned are cache hits, and therefore complete as quickly as possible.

Cache miss

A memory access that cannot be processed at high speed because the instruction/data it addresses is not in the cache and a main memory access is required.

Cache set

A cache set is a group of cache lines (or blocks). A set contains all the ways that can be addressed with the same index. The number of cache sets is always a power of two.

See also Cache terminology diagram on the last page of this glossary.

Cache way

A group of cache lines (or blocks). It is 2 to the power of the number of index bits in size.

See also Cache terminology diagram on the last page of this glossary.

Cast out

See Victim.

Clean

A cache line that has not been modified while it is in the cache is said to be clean. To clean a cache is to write dirty cache entries into main memory. If a cache line is clean, it is not written on a cache miss because the next level of memory contains the same data as the cache.

See also Dirty.

Coherency

See Memory coherency.

Copy back

See Write-back.

Core

A core is that part of a processor that contains the ALU, the datapath, the general-purpose registers, the Program Counter, and the instruction decode and control circuitry.

Data Abort

An indication from a memory system to a core that it must halt execution of an attempted illegal memory access. A Data Abort is attempting to access invalid data memory.

See also Abort, External Abort, and Prefetch Abort.

Data Cache

A block of on-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used data. This is done to greatly increase the average speed of memory accesses and so to increase processor performance.

Direct-mapped cache
A one-way set-associative cache. Each cache set consists of a single cache line, so cache look-up selects and checks a single cache line.

Dirty
A cache line in a Write-Back cache that has been modified while it is in the cache is said to be dirty. A cache line is marked as dirty by setting the dirty bit. If a cache line is dirty, it must be written to memory on a cache miss because the next level of memory contains data that has not been updated. The process of writing dirty data to main memory is called cache cleaning.

See also Clean.

Drain
Force the contents of the write buffer to be written to main memory.

Endianness
Byte ordering. The scheme that determines the order in which successive bytes of a data word are stored in memory. An aspect of the system's memory mapping.

See also Little-endian and Big-endian

Exception
A fault or error event that is considered serious enough to require that program execution is interrupted. Examples include attempting to perform an invalid memory access, external interrupts, and undefined instructions. When an exception occurs, normal program flow is interrupted and execution is resumed at the corresponding interrupt vector. This contains the first instruction of the interrupt handler to deal with the exception.

Fully-associative cache
A cache that has just one cache set that consists of the entire cache. The number of cache entries is the same as the number of cache ways.

See also Direct-mapped cache.

Index
See Cache index.

Instruction Cache
A block of on-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used instructions. This is done to increase the average speed of memory accesses and therefore to increase processor performance.

Invalidate
To mark a cache line as being not valid by clearing the valid bit. This must be done whenever the line does not contain a valid cache entry. For example, after a cache flush all lines are invalid.

Line
See Cache line.

Little-endian
Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also Big-endian and Endianness.

Little-endian memory
Memory in which: - a byte or halfword at a word-aligned address is the least significant byte or halfword within the word at that address - a byte at a halfword-aligned address is the least significant byte within the halfword at that address.

See also Big-endian memory.

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

PA
See Physical Address.

Physical Address (PA)
The MMU performs a translation on *Modified Virtual Addresses* (MVA) to produce the *Physical Address* (PA) which is given to AHB to perform an external access. The PA is also stored in the data cache to avoid the necessity for address translation when data is cast out of the cache.

SBO
See Should Be One.

SBZ
See Should Be Zero.

Set-associative cache
In a set-associative cache, lines can only be placed in the cache in locations that correspond to the modulo division of the memory address by the number of sets. If there are *n* ways in a cache, the cache is termed *n*-way set-associative. The set-associativity can be any number greater than or equal to 1 and is not restricted to being a power of two.

Should Be One (SBO)
Should be written as 1 (or all 1s for bit fields) by software. Writing a 0 produces Unpredictable results.

Should Be Zero (SBZ)
Should be written as 0 (or all 0s for bit fields) by software. Writing a 1 produces Unpredictable results.

Tag
The upper portion of a block address used to identify a cache line within a cache. The block address from the CPU is compared with each tag in a set in parallel to determine if the corresponding line is in the cache. If it is it is said to be a cache hit and the line can be fetched from cache. If the block address does not correspond to any of the tags it is said to be a cache miss and the line must be fetched from the next level of memory.

TLB *See* Translation Lookaside Buffer.

Translation Lookaside Buffer (TLB)
A cache of recently used page table entries that avoid the overhead of page table walking on every memory access. Part of the Memory Management Unit.

Victim A cache line, selected to be discarded to make room for a replacement cache line that is required as a result of a cache miss. The way in which the victim is selected for eviction is processor-specific. A victim is also known as a cast out.

Way *See* Cache way.

WB *See* Write-back.

Word A 32-bit data item.

Write-back (WB) In a write-back cache, data is only written to main memory when it is forced out of the cache on line replacement following a cache miss. Otherwise, writes by the processor only update the cache. (Also known as copyback).

Write buffer A block of high-speed memory, arranged as a FIFO buffer, between the Data Cache and main memory, whose purpose is to optimize stores to main memory. Each entry in the write buffer can contain the address of a data item to be stored to main memory, the data for that item, and a sequential bit that indicates if the next store is sequential or not.

Write completion
The memory system indicates to the CPU that a write has been completed at a point in the transaction where the memory system is able to guarantee that the effect of the write is visible to all processors in the system. This is not the case if the write is associated with a memory synchronization primitive, or is to a Device or Strongly Ordered region. In these cases the memory system might only indicate completion of the write when the access has affected the state of the target, unless it is impossible to distinguish between having the effect of the write visible and having the state of target updated. This stricter requirement for some types of memory ensures that any side-effects of the memory access can be guaranteed by the processor to have taken place. You can use this to prevent the starting of a subsequent operation in the program order until the side-effects are visible.

Write-through
In a write-through cache, data is written to main memory at the same time as the cache is updated.

Cache terminology
The diagram below illustrates the following cache terminology:

- block address
- cache line
- cache set

- cache way
- index
- tag.