

PrimeCell[®] Level 2 Cache Controller (PL310)

Revision: r2p0

Technical Reference Manual



PrimeCell Level 2 Cache Controller (PL310)

Technical Reference Manual

Copyright © 2007, 2008 ARM Limited. All rights reserved.

Release Information

Change history			
Date	Issue	Confidentiality	Change
30 November 2007	A	Non-Confidential	First release for r0p0
04 April 2008	B	Non-Confidential	First release for r1p0
19 December 2008	C	Non-Confidential Unrestricted Access	First release for r2p0

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.

Unrestricted Access is an ARM internal classification.

Product Status

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

Web Address

<http://www.arm.com>

Contents

PrimeCell Level 2 Cache Controller (PL310)

Technical Reference Manual

Preface

About this manual	xii
Feedback	xvi

Chapter 1

Introduction

1.1 About the PrimeCell level 2 cache controller (PL310)	1-2
1.2 Typical system configuration	1-7
1.3 Product revisions	1-9

Chapter 2

Functional Overview

2.1 Cache configurability	2-2
2.2 AXI master and slave interfaces	2-4
2.3 Cache operation	2-13
2.4 RAM interfaces	2-27
2.5 Implementation details	2-39
2.6 Power modes	2-50

Chapter 3

Programmers Model

3.1 About this programmers model	3-2
3.2 Register summary	3-4

3.3	Register descriptions	3-8
-----	-----------------------------	-----

Appendix A **Signal Descriptions**

A.1	Clock and reset	A-2
A.2	Configuration	A-3
A.3	Slave and master ports	A-4
A.4	RAM interface	A-12
A.5	Cache event monitoring	A-15
A.6	Cache interrupt	A-16
A.7	MBIST interface	A-17

Appendix B **AC Parameters**

B.1	Reset and configuration signal timing parameters	B-2
B.2	Slave port 0 I/O signal timing parameters	B-3
B.3	Slave port 1 I/O signal timing parameters	B-5
B.4	Master port 0 I/O signal timing parameters	B-7
B.5	Master port 1 I/O signal timing parameters	B-9
B.6	RAMs signal timing parameters	B-11
B.7	Event monitor signal timing parameters	B-14
B.8	Cache interrupt ports signal timing parameters	B-15
B.9	MBIST interface signal timing parameters	B-16

Appendix C **Timing Diagrams**

C.1	Single read hit transaction	C-2
C.2	Single read miss transaction	C-3
C.3	Single noncacheable read transaction	C-4
C.4	Outstanding read hit transaction	C-5
C.5	Hit under miss read transactions	C-6
C.6	Single bufferable write transaction	C-7
C.7	Single nonbufferable write transaction	C-8

Appendix D **Revisions**

Glossary

List of Tables

PrimeCell Level 2 Cache Controller (PL310)

Technical Reference Manual

	Change history	ii
Table 1-1	Typical memory sizes and access times	1-2
Table 1-2	RTL options	1-5
Table 1-3	Master port transactions for a two master port system	1-8
Table 2-1	Cache controller cache configurability	2-2
Table 2-2	AXI master interface attributes	2-4
Table 2-3	AXI slave interface attributes	2-5
Table 2-4	Master port ID values for writes	2-8
Table 2-5	Master port ID values for reads	2-8
Table 2-6	Exported master ports AXI control signals	2-9
Table 2-7	Noncacheable and nonbufferable AXI transactions	2-13
Table 2-8	Bufferable only AXI transactions	2-13
Table 2-9	Cacheable but do not allocate AXI transactions	2-13
Table 2-10	Cacheable write-through, allocate on read AXI transactions	2-14
Table 2-11	Cacheable write-back, allocate on read AXI transactions	2-14
Table 2-12	Cacheable write-through, allocate on write AXI transactions	2-15
Table 2-13	Cacheable write-back, allocate on write AXI transactions	2-15
Table 2-14	Cacheable write-through, allocate on read and write AXI transactions	2-16
Table 2-15	Cacheable write-back, allocate on read and write AXI transactions	2-16
Table 2-16	AWCACHE and ARCACHE definitions	2-17
Table 2-17	MP4 system lockdown register definitions	2-24

Table 2-18	MPI plus APC system lockdown register definitions.	2-26
Table 2-19	RAM clock enables	2-34
Table 2-20	Error responses for all combinations of L3 access	2-43
Table 2-21	Event pins	2-44
Table 2-22	Interrupts	2-45
Table 2-23	Cacheable read requests on AXI slave ports	2-47
Table 2-24	Write-through/write-back write access from store buffer	2-47
Table 2-25	AXI M0 and AXI M1 masters or store buffer allocation requests	2-48
Table 2-26	Clean maintenance operation cases	2-48
Table 2-27	Invalidate maintenance operation cases	2-49
Table 2-28	Clean and Invalidate maintenance operation cases	2-49
Table 3-1	Cache controller register map	3-4
Table 3-2	Summary of cache controller registers	3-5
Table 3-3	reg0_cache_id Register bit assignments	3-8
Table 3-4	reg0_cache_type Register bit assignments	3-9
Table 3-5	reg1_control Register bit assignments	3-11
Table 3-6	reg1_aux_control Register bit assignments	3-13
Table 3-7	reg1_tag_ram_control and reg1_data_ram_control Register bit assignments	3-16
Table 3-8	reg2_ev_counter_ctrl Register bit assignments	3-18
Table 3-9	reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments	3-19
Table 3-10	reg2_ev_counter0 and reg2_ev_counter1 Register bit assignments	3-20
Table 3-11	reg2_int_mask Register bit assignments	3-22
Table 3-12	Masked Interrupt Status Register bit assignments	3-23
Table 3-13	reg2_int_raw_status Register bit assignments	3-24
Table 3-14	reg2_int_clear Register bit assignments	3-25
Table 3-15	Maintenance operations	3-26
Table 3-16	Cache maintenance operations	3-29
Table 3-17	Cache lockdown	3-34
Table 3-18	Lockdown by Line Enable Register bit assignments	3-35
Table 3-19	Unlock All Lines Register bit assignments	3-35
Table 3-20	Data Lockdown 0 Register, offset 0x900	3-35
Table 3-21	Instruction Lockdown 0 Register, offset 0x904	3-35
Table 3-22	Data Lockdown 1 Register, offset 0x908	3-36
Table 3-23	Instruction Lockdown 1 Register, offset 0x90C	3-36
Table 3-24	Data Lockdown 2 Register, offset 0x910	3-36
Table 3-25	Instruction Lockdown 2 Register, offset 0x914	3-36
Table 3-26	Data Lockdown 3 Register, offset 0x918	3-36
Table 3-27	Instruction Lockdown 3 Register, offset 0x91C	3-36
Table 3-28	Data Lockdown 4 Register, offset 0x920	3-37
Table 3-29	Instruction Lockdown 4 Register, offset 0x924	3-37
Table 3-30	Data Lockdown 5 Register, offset 0x928	3-37
Table 3-31	Instruction Lockdown 5 Register, offset 0x92C	3-37
Table 3-32	Data Lockdown 6 Register, offset 0x930	3-37
Table 3-33	Instruction Lockdown 6 Register, offset 0x934	3-37
Table 3-34	Data Lockdown 7 Register, offset 0x938	3-38
Table 3-35	Instruction Lockdown 7 Register, offset 0x93C	3-38
Table 3-36	Address Filtering Start Register bit assignments	3-39

Table 3-37	Address Filtering End Register bit assignments	3-40
Table 3-38	Debug Control Register bit assignments	3-41
Table 3-39	Prefetch Offset Register bit assignments	3-43
Table A-1	Clock and reset signals	A-2
Table A-2	Configuration signals	A-3
Table A-3	Slave port 0 signals	A-4
Table A-4	Slave port 1 signals	A-6
Table A-5	Master port 0 signals	A-8
Table A-6	Master port 1 signals	A-10
Table A-7	Data RAM interface signals	A-12
Table A-8	Tag RAM interface	A-13
Table A-9	Cache event monitoring signals	A-15
Table A-10	Cache Interrupt signals	A-16
Table A-11	MBIST interface signals	A-17
Table B-1	Reset and configuration	B-2
Table B-2	Slave port 0 I/O	B-3
Table B-3	Slave port 1 I/O	B-5
Table B-4	Master port 0 I/O	B-7
Table B-5	Master port 1 I/O	B-9
Table B-6	Data RAM	B-11
Table B-7	Tag RAM	B-12
Table B-8	Event monitor	B-14
Table B-9	Cache interrupt ports	B-15
Table B-10	MBIST interface signal	B-16
Table D-1	Differences between issue B and issue C	D-1

List of Figures

PrimeCell Level 2 Cache Controller (PL310)

Technical Reference Manual

	Key to timing diagram conventions	xiv
Figure 1-1	Top level diagram	1-3
Figure 1-2	Example cache controller interfaced to an ARM processor	1-7
Figure 2-1	CLKEN used to drive cache controller inputs in case of integer clock ratio	2-6
Figure 2-2	Clock enable usage model for 1.5:1 clock ratio in master port	2-7
Figure 2-3	Driven by cortex a9 cluster with 4 cpus	2-24
Figure 2-4	Driven by cortex a9 cluster with 1 CPU and ACP	2-25
Figure 2-5	Data RAM organization for 16 ways	2-28
Figure 2-6	Tag RAM organization for a 16-way 256KB L2 cache, with parity, with lockdown by line 2-30	
Figure 2-7	Data parity RAM organization	2-31
Figure 2-8	Data RAM address bus format for 16 ways	2-31
Figure 2-9	Tag RAM running at slower frequency	2-35
Figure 2-10	Tag RAM clock gating	2-35
Figure 2-11	Tag RAM setup latency	2-36
Figure 2-12	Tag RAM read access latency	2-36
Figure 2-13	Tag RAM write access latency	2-37
Figure 2-14	MBIST interface for 16-way implementation, with parity, without lockdown by line	2-38
Figure 2-15	Parity and RAM error support for a 16-way implementation	2-46
Figure 3-1	reg0_cache_id Register bit assignments	3-8
Figure 3-2	reg0_cache_type Register bit assignments	3-9

Figure 3-3	reg1_control Register bit assignments	3-11
Figure 3-4	reg1_aux_control Register bit assignments	3-12
Figure 3-5	reg1_tag_ram_control and reg1_data_ram_control Register bit assignments	3-15
Figure 3-6	reg2_ev_counter_ctrl Register bit assignments	3-18
Figure 3-7	reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments	3-19
Figure 3-8	Interrupt Register bit assignments	3-21
Figure 3-9	Physical address format	3-27
Figure 3-10	Index/way format	3-27
Figure 3-11	Way format	3-28
Figure 3-12	Address Filtering Start Register bit assignments	3-39
Figure 3-13	Address Filtering End Register bit assignments	3-40
Figure 3-14	Debug Control Register bit assignments	3-41
Figure 3-15	Prefetch Offset Register bit assignments	3-42
Figure C-1	Single read hit transaction	C-2
Figure C-2	Single read miss transaction	C-3
Figure C-3	Single noncacheable read transaction	C-4
Figure C-4	Outstanding read hit transaction	C-5
Figure C-5	Hit under miss read transaction	C-6
Figure C-6	Single bufferable write transaction	C-7
Figure C-7	Single nonbufferable write transaction	C-8

Preface

This preface introduces the *PrimeCell Level 2 Cache Controller (PL310) Revision r1p0 Technical Reference Manual*. It contains the following sections:

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

About this manual

This is the *Technical Reference Manual* (TRM) for the PrimeCell Level 2 Cache Controller (PL310). In this manual the generic term cache controller means the PrimeCell Level 2 Cache Controller.

Product revision status

The *rn**pn* identifier indicates the revision status of the product described in this manual, where:

rn Identifies the major revision of the product.

pn Identifies the minor revision or modification status of the product.

Intended audience

This manual is written for hardware and software engineers implementing the PrimeCell Level 2 Cache Controller into ASIC designs. It provides information to enable designers to integrate the device into a target system as quickly as possible.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the cache controller.

Chapter 2 *Functional Overview*

Read this for a description of a functional overview and the functional operation of the cache controller.

Chapter 3 *Programmers Model*

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

Appendix A *Signal Descriptions*

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

Appendix B *AC Parameters*

Read this for a description of the AC signal timing parameters

Appendix C *Timing Diagrams*

Read this for a description of cache controller timing diagrams.

Appendix D Revisions

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

Glossary

Read this for definitions of terms used in this manual.

Conventions

Conventions that this manual can use are described in:

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

Typographical

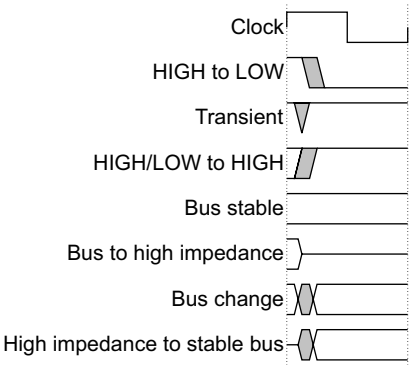
The typographical conventions are:

<i>italic</i>	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.
<code>monospace</code>	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u><code>monospace</code></u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<i><code>monospace italic</code></i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
<code>monospace bold</code>	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>

Timing diagrams

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

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



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means: <ul style="list-style-type: none">• HIGH for active-HIGH signals• LOW for active-LOW signals.
Lower-case n	At the start or end of a signal name denotes an active-LOW signal.
Prefix A	Denotes global <i>Advanced eXtensible Interface</i> (AXI) signals.
Prefix AF	Denotes <i>Advanced Trace Bus</i> (ATB) flush control signals.
Prefix AR	Denotes AXI read address channel signals.
Prefix AT	Denotes ATB data flow 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 <i>Advanced High-performance Bus</i> (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.

Further reading

This section lists publications by ARM and by third parties.

See <http://infocenter.arm.com> for access to ARM documentation.

ARM publications

This document contains information that is specific to the cache controller. See the following documents for other relevant information:

- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *ARM Architecture Reference Manual ARM v7-A and ARM v7-R* (ARM DDI 0406)
- *ARM PrimeCell Level 2 MBIST Controller (PL310) Technical Reference Manual* (ARM DDI 0402)
- *ARM PrimeCell Level 2 Cache Controller (PL310) Implementation Guide* (ARM DII 0045)
- *ARM Cortex™-A9 Technical Reference Manual* (ARM DDI 0388).

Feedback

ARM welcomes feedback on the 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.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms if appropriate.

Feedback on this manual

If you have any comments on this manual, send e-mail to errata@arm.com. Give:

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

ARM also welcomes general suggestions for additions and improvements.

Chapter 1

Introduction

This chapter introduces the PrimeCell level 2 cache controller (PL310) and its features. It contains the following sections:

- *About the PrimeCell level 2 cache controller (PL310)* on page 1-2
- *Typical system configuration* on page 1-7
- *Product revisions* on page 1-9.

1.1 About the PrimeCell level 2 cache controller (PL310)

The addition of an on-chip secondary cache, also referred to as a Level 2 or L2 cache, is a recognized method of improving the performance of ARM-based systems when significant memory traffic is generated by the processor. By definition a secondary cache assumes the presence of a Level 1 or primary cache, closely coupled or internal to the processor.

Memory access is fastest to L1 cache, followed closely by L2 cache. Memory access is typically significantly slower with L3 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	Typical size	Typical access time
Processor registers	128B	1 cycle
On-chip L1 cache	32KB	1-2 cycles
On-chip L2 cache	256KB	8 cycles
Main memory (L3) dynamic RAM	MB or GB ^a	30-100 cycles
Back-up memory (hard disk) (L4)	MB or GB	> 500 cycles

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

The cache controller features:

- TrustZone architecture for enhanced OS security.
- Slave and master AMBA AXI interfaces designed for high performance systems.

The cache controller is a unified, physically addressed, physically tagged cache with up to 16 ways. You can lock the replacement algorithm on a way basis, enabling the associativity to be reduced from 16-way down to one-way, direct mapped.

The cache controller does not have snooping hardware to maintain coherency between caches, so you must maintain coherency by software.

Figure 1-1 on page 1-3 shows a top level diagram of the cache controller.

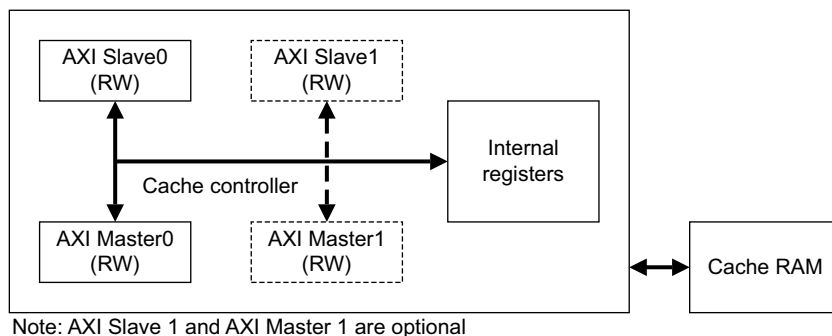


Figure 1-1 Top level diagram

1.1.1 Features

The cache controller has the following features:

- Physically addressed and physically tagged.
- Lockdown format C supported, for data and instructions.

———— Note ————

Lockdown format C is also known as way locking.

- Lockdown by line supported.
- Lockdown by master id supported.
- L2 cache available size can be 16KB to 8MB, depending on configuration and the use of the lockdown registers.
- Direct mapped to 16-way associativity, depending on the configuration and the use of lockdown registers. The associativity is RTL configurable as 8 or 16.
- Fixed line length of 32 bytes (eight words or 256 bits).
- Interface to data RAM is byte writable.
- Supports all of the AXI cache modes:
 - write-through and write-back
 - read allocate, write allocate, read and write allocate.
- Force write allocate option to always have cacheable writes allocated to L2 cache, for processors not supporting this mode.

- Normal memory non-cacheable shared reads are treated as cacheable non-allocatable. Normal memory non-cacheable shared writes are treated as cacheable write-through no write-allocate. There is an option, Shared Override, to override this behavior. For details on the Shared attribute see *Shared attribute* on page 2-19.
- TrustZone support, with the following features:
 - *Non-Secure* (NS) tag bit added in tag RAM and used for lookup in the same way as an address bit. The NS-tag bit is added in all buffers.
 - NS bit in Tag RAM used to determine security level of evictions to L3.
 - Restrictions for NS accesses for control, configuration, and maintenance registers to restrict access to secure data.
- Critical word first linefill supported.
- Pseudo-Random victim selection policy. You can make this deterministic with use of lockdown registers.
- Four 256-bit *Line Fill Buffers* (LFBs), shared by the master ports. These buffers capture linefill data from main memory, waiting for a complete line before writing to L2 cache memory.
- Two 256-bit *Line Read Buffers* (LRBs) for each slave port. These buffers hold a line from the L2 memory in case of cache hit.
- Three 256-bit *Eviction Buffers* (EBs). These buffers hold evicted lines from the L2 cache, to be written back to main memory.
- Three 256-bit *Store Buffers* (STBs). These buffers hold bufferable writes before their draining to main memory, or the L2 cache. They enable multiple writes to the same line to be merged.
- Supports outstanding accesses on slave and master ports.
- Option to select one or two master ports.
- Option to select one or two slave ports. If only one slave is supported, only one master is configured.
- Software option to enable exclusive cache configuration, see *Cache operation* on page 2-13 for more information.
- Prefetching capability. See *Auxiliary Control Register* on page 3-12 for more information.
- Support for integer (1:1, 2:1,...) and half-integer (1.5:1, 2.5:1,...) clock ratios controlled by clock enable inputs on slave and master ports.

- Wait, latency, clock enable, parity, and error support at the RAM interfaces.
- MBIST support.
- L2 cache event monitoring. Event signals are exported if you want to use these in conjunction with an event monitoring block. Event monitoring is also offered in the cache controller with two programmable 32-bit counters. Secure event and performance signals are only available when the signal on the **SPNIDEN** pin is configured HIGH.
- Configuration registers accessible using address decoding in the slave ports.
- Address filtering in the master ports enabling redirection of a certain address range to one master port while all other addresses are redirected to the other one.

A number of RTL options enable you to implement the RTL with different features present or absent, as shown in Table 1-2.

Table 1-2 RTL options

Feature	RTL option
16-way associativity	p1310_16_WAYS
Number of slave ports	p1310_S1
Number of master ports	p1310_M1 (requires p1310_S1)
Parity	p1310_PARITY
Address filtering	p1310_ADDRESS_FILTERING (requires p1310_M1)
Lockdown by master	p1310_LOCKDOWN_BY_MASTER
Lockdown by line	p1310_LOCKDOWN_BY_LINE
Slave AXI ID width	p1310_AXI_ID_MAX
RAM latencies	p1310_TAG_SETUP_LAT p1310_TAG_READ_LAT p1310_TAG_WRITE_LAT p1310_DATA_SETUP_LAT p1310_DATA_READ_LAT p1310_DATA_WRITE_LAT

Cache configurability on page 2-2 shows how you can use these RTL options to configure the cache controller.

Note

Before synthesis you must define these options in the pl310_defs.v verilog file.

The cache controller is configured using memory-mapped registers, rather than using CP15 instructions. See Chapter 3 *Programmers Model* for more information.

The cache controller is designed to work with 64-bit AXI masters. No particular primary cache architecture is assumed.

1.2 Typical system configuration

The cache controller works efficiently with ARM processors that implement AXI interfaces. It directly interfaces on the data and instruction interface. The internal pipelining of the cache controller is optimized to enable the processors to operate at the same clock frequency.

The cache controller supports:

- One or two read/write 64-bit slave ports for interfacing with data and instruction interfaces.
- One or two read/write 64-bit master ports for interfacing with L3 memory system.

Figure 1-2 shows an example of a cache controller with two slave ports and two master ports interfaced to an ARM processor.

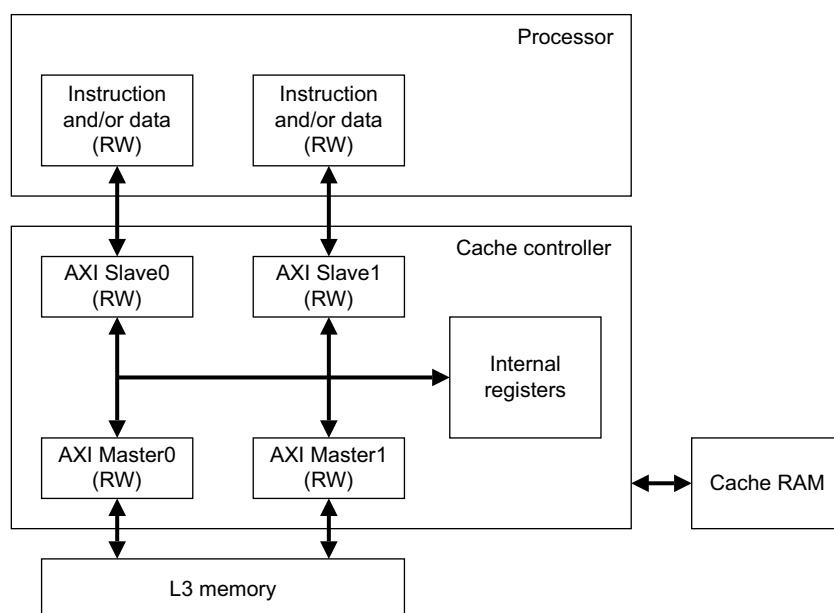


Figure 1-2 Example cache controller interfaced to an ARM processor

You can configure the cache controller to use one or two master ports. Table 1-3 shows what each master port is used for.

Table 1-3 Master port transactions for a two master port system

Master port 0	Master port 1
Non-cacheable reads from S0	Non-cacheable reads from S1
Linefills from S0	Linefills from S1
Write allocations reads from STB	Write allocations reads from STB
Non-bufferable writes from S0	Non-bufferable writes from S1
Bufferable writes from STB	Bufferable writes from STB
Evictions from EB	Evictions from EB

———— **Note** ————

- Table 1-3 does not take address filtering into account. If address filtering is implemented and enabled:
 - master port 1 deals with all transactions from **S0**, **S1**, STB, and EB targeting the defined address range
 - master port 0 deals with all other transactions.
- In a one master port system, master port 1 is not implemented. All master port 0 transactions apply to both **S0** and **S1**.

1.3 Product revisions

This section summarizes the differences in functionality between the releases of the cache controller:

r0p0-r1p0 The main differences between these versions are:

- New RAM latency scheme introduced and new configuration registers to support this scheme.
- New RAM clocking scheme introduced enabling RAMs to be run at a lower frequency to the cache controller.
- AXI slave and master interface attributes changed.
- Device writes can go into store buffer.
- Additional **ARUSERSx/AWUSERSx** signals.

r1p0-r2p0 The main differences between these versions are:

- New behavior linked to the Shared attribute.
- New configuration for supporting full address hazard checking.
- Performance improvements linked to the Cortex-A9 processor.
- New AXI ID dedicated to Device writes from store buffer.

Chapter 2

Functional Overview

This chapter describes the cache controller and its features. It contains the following sections:

- *Cache configurability* on page 2-2
- *AXI master and slave interfaces* on page 2-4
- *Cache operation* on page 2-13
- *RAM interfaces* on page 2-27
- *Implementation details* on page 2-39
- *Power modes* on page 2-50.

2.1 Cache configurability

Table 2-1 shows how you can configure the cache controller.

Table 2-1 Cache controller cache configurability

Feature	Enabled by	Range of options	Default value	Default option
Cache way size	Register or WAYSIZES[2:0] input	16KB, 32KB, 64KB, 128KB, 256KB, 512KB	001	16KB
Number of cache ways	Register or ASSOCIATIVITY input	8, 16 ^a	0	8 ways
RAM latencies	Register or verilog `define	1, 2, 3, 4, 5, 6, 7, 8	111	8 cycles of latency
Slave port 1 present	Verilog `define p1310_S1	Commented or uncommented	Commented	No slave port 1
Master port 1 present	Verilog `define p1310_M1	Commented or uncommented	Commented	No master port 1
Parity logic	Verilog `define p1310_PARITY	Commented or uncommented	Commented	No parity logic
Lockdown by master	Verilog `define p1310_LOCKDOWN_BY_MASTER	Commented or uncommented	Commented	No lockdown by master
Lockdown by line	Verilog `define p1310_LOCKDOWN_BY_LINE	Commented or uncommented	Commented	No lockdown by line
AXI ID width on slave ports	Verilog `define p1310_AXI_ID_MAX <value>	>=1	5	6 AXI ID bits on slave ports and 8 on master ports
Address filtering	Verilog `define p1310_ADDRESS_FILTERING	Commented or uncommented	Commented	No address filtering logic
Full address hazard checking	Verilog `define p1310_FULL_ADDR_HAZARD	Commented (hazard checking done on 16 address bits, that is bits [20:5]) or uncommented (hazard checking done on the whole address, that is bits [31:5])	Commented	Hazard checking done on 16 address bits, that is bits [20:5]

a. 16-way associativity must be enabled using the p1310_16_WAYS verilog `define.

Note

If you configure a single slave port, **AXI S0**, you must only configure a single master port, **AXI M0**. If you configure address filtering, you must configure master port **AXI M1**.

2.2 AXI master and slave interfaces

- This section describes:
- *AXI master and slave interface attributes*
 - *Clock enable usage model in the cache controller AXI interfaces* on page 2-6
 - *Master and slave port IDs* on page 2-7
 - *Exported AXI control* on page 2-9
 - *AXI locked and exclusive accesses* on page 2-11.

2.2.1 AXI master and slave interface attributes

Table 2-2 shows the AXI master interface attributes.

Table 2-2 AXI master interface attributes

Configuration	Attribute	Value
Two master ports	Write issuing capability	12, consisting of: <ul style="list-style-type: none">• 6 evictions• 6 writes from store buffer.
	Read issuing capability	11, consisting of: <ul style="list-style-type: none">• 8 reads from slave ports (4 per master port)• 3 reads from store buffer.
	Combined issuing capability	23
	Write interleave capability	2 (1 per master port)
	Write ID width	Defined by Write ID width on slave ports.
	Read ID width	Defined by Read ID width on slave ports.

Table 2-2 AXI master interface attributes (continued)

Configuration	Attribute	Value
One master port	Write issuing capability	12, consisting of: <ul style="list-style-type: none"> • 6 evictions • 6 writes from store buffer.
	Read issuing capability	7, consisting of: <ul style="list-style-type: none"> • 4 reads from slave port(s) • 3 reads from store buffer.
	Combined issuing capability	19
	Write interleave capability	1
	Write ID width	Defined by Write ID width on slave port(s).
	Read ID width	Defined by Read ID width on slave port(s).

Table 2-3 shows the AXI slave interface attributes.

Table 2-3 AXI slave interface attributes

Configuration	Attribute	Value
Two slave ports and two master ports	Write acceptance capability	8 (4 per slave port)
	Read acceptance capability	16 (8 per slave port)
	Combined acceptance capability	24
	Write interleave depth	2 (1 per slave port)
	Read data reorder depth	16
	Write ID width	Parameterizable, defined by p1310_AXI_ID_MAX, default is 6
	Read ID width	Parameterizable, defined by p1310_AXI_ID_MAX, default is 6

Table 2-3 AXI slave interface attributes (continued)

Configuration	Attribute	Value
Two slave ports and one master port	Write acceptance capability	8 (4 per slave port)
	Read acceptance capability	12 (6 per slave port)
	Combined acceptance capability	20
	Write interleave depth	2 (1 per slave port)
	Read data reorder depth	12
	Write ID width	Parameterizable, defined by p1310_AXI_ID_MAX, default is 6
	Read ID width	Parameterizable, defined by p1310_AXI_ID_MAX, default is 6
One slave port and one master port	Write acceptance capability	3
	Read acceptance capability	8
	Combined acceptance capability	12
	Write interleave depth	1
	Read data reorder depth	8
	Write ID width	Parameterizable, defined by p1310_AXI_ID_MAX, default is 6
	Read ID width	Parameterizable, defined by p1310_AXI_ID_MAX, default is 6

2.2.2 Clock enable usage model in the cache controller AXI interfaces

The cache controller receives one clock, **CLK**. The AXI slave and master ports receive clock enable pins that enable you to define different clock ratios, which can be integer or half-integer (1.5:1, 2.5:1, and 3.5:1). Each master and slave receives one clock enable for its AXI inputs and one for its AXI outputs.

For integer clock ratios, the clock enable **CLKEN** of the system must be used to drive both cache controller inputs, **INCLKEN** and **OUTCLKEN**, as shown in Figure 2-1.



Figure 2-1 CLKEN used to drive cache controller inputs in case of integer clock ratio

Figure 2-2 on page 2-7 shows how the different clock enable signals can be generated to support half integer clock ratios. In the figure, **ACLK** is the clock of the L3 AXI system.

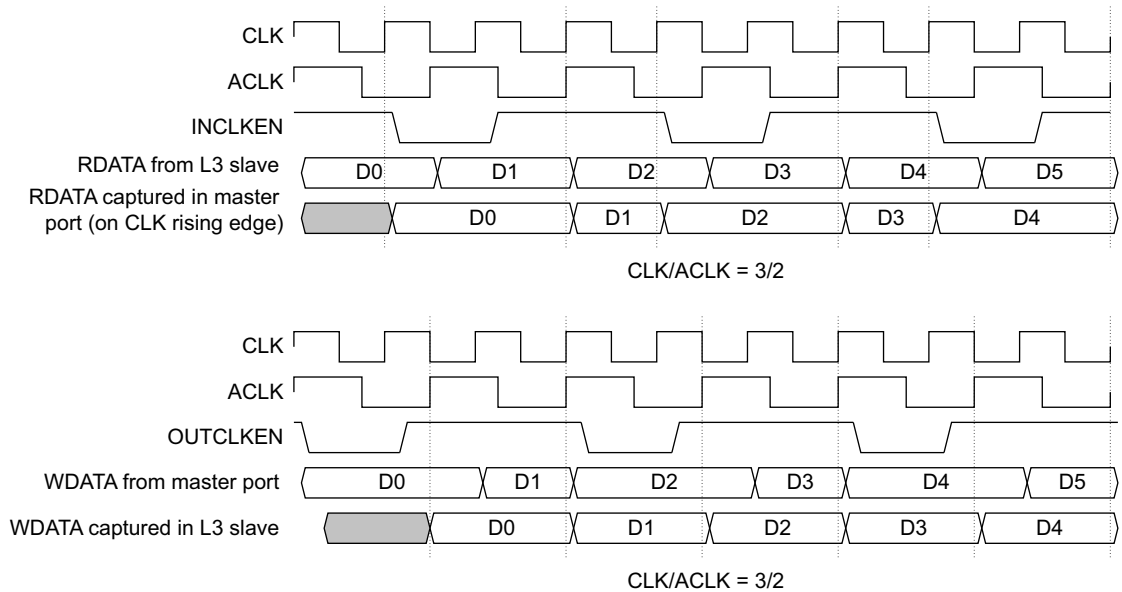


Figure 2-2 Clock enable usage model for 1.5:1 clock ratio in master port

See Table A-1 on page A-2 for a description of the clock enables. These clock enables enable the cache controller to communicate with AXI components that run at slower frequencies.

In each of these AXI interfaces, the clock enable is used as follows:

- Inputs are sampled on rising edges of **CLK** only when **INCLKEN** is HIGH.
- Outputs are updated on rising edges of **CLK** only when **OUTCLKEN** is HIGH.

2.2.3 Master and slave port IDs

The AXI ID width on the slave and master ports depends on the value of parameter `p1310_AXI_ID_MAX` that has a default value of 5. The AXI ID width is `[`p1310_AXI_ID_MAX:0]` on the slave ports and `[`p1310_AXI_ID_MAX+2:0]` on the master ports. This section assumes that the default value for `p1310_AXI_ID_MAX` is 5.

The number of bits for master and slave port IDs are as follows:

- Slave **S0**: 6 bits [5:0], signals **AWIDS0**, **ARIDS0**, **WIDS0**, **BIDS0**, and **RIDS0**
- Slave **S1**: 6 bits [5:0], signals **AWIDS1**, **ARIDS1**, **WIDS1**, **BIDS1**, and **RIDS1**
- Master **M0**: 8 bits [7:0], signals **AWIDM0**, **ARIDM0**, **WIDM0**, **BIDM0**, and **RIDM0**

- Master **M1**: 8 bits [7:0], signals **AWIDM1**, **ARIDM1**, **WIDM1**, **BIDM1**, and **RIDM1**.

Table 2-4 shows the format of the identifications that are exported on the master ports for writes.

Table 2-4 Master port ID values for writes

Master ID buses	Access types	ID value (Verilog)
AWIDMx, WIDMx, BIDMx	Non-bufferable write from slave 0	{AWIDS0, 00}
	Non-bufferable write from slave 1	{AWIDS1, 10}
	Eviction from slot 0	{{`p1310_AXI_ID_MAX-1{0}}, 0011}
	Eviction from slot 1	{{`p1310_AXI_ID_MAX-1{0}}, 0111}
	Eviction from slot 2	{{`p1310_AXI_ID_MAX-1{0}}, 1011}
	Device write from store buffer	{{`p1310_AXI_ID_MAX-1{0}}, 1101}
	Write from store buffer slot 0	{{`p1310_AXI_ID_MAX-1{0}}, 0001}
	Write from store buffer slot 1	{{`p1310_AXI_ID_MAX-1{0}}, 0101}
	Write from store buffer slot 2	{{`p1310_AXI_ID_MAX-1{0}}, 1001}

Table 2-5 shows the format of the identifications that are exported on the master ports for reads.

Table 2-5 Master port ID values for reads

Master ID buses	Access types	ID value (Verilog)
ARIDMx, RIDMx	Non-cacheable or linefill from slave 0	{ARIDS0, 00}
	Non-cacheable or linefill from slave 1	{ARIDS1, 10}
	Read from store buffer slot 0	{{`p1310_AXI_ID_MAX-1{0}}, 0101}
	Read from store buffer slot 1	{{`p1310_AXI_ID_MAX-1{0}}, 1001}
	Read from store buffer slot 2	{{`p1310_AXI_ID_MAX-1{0}}, 1101}
	Prefetch from linefill buffer slot 0	{{`p1310_AXI_ID_MAX-1{0}}, 0011}

Table 2-5 Master port ID values for reads (continued)

Master ID buses	Access types	ID value (Verilog)
	Prefetch from linefill buffer slot 1	{{`p1310_AXI_ID_MAX-1{0}}, 0111}
	Prefetch from linefill buffer slot 2	{{`p1310_AXI_ID_MAX-1{0}}, 1011}
	Prefetch from linefill buffer slot 3	{{`p1310_AXI_ID_MAX-1{0}}, 1111}

The L2 CC returns these ID bits with the data on the RID lines for a read transaction and accompanies the write response signal as BID for a write transaction.

2.2.4 Exported AXI control

Table 2-6 provides information on the AXI control signals on the master ports of the cache controller.

Table 2-6 Exported master ports AXI control signals

Access type	Master control signals buses	Value (Verilog)
Noncacheable read transactions, nonbufferable write transactions, or cache disabled	AWCACHEx/ARCACHEx AWPROTMx/ARPROTMx AWLOCKMx/ARLOCKMx AWADDRMx/ARADDRMx AWLENMx/ARLENMx AWSIZEx/ARSIZEx AWBURSTMx/ARBURSTMx	All as original transaction
Linefills associated with a cacheable read transaction	ARCACHEx	As original transaction
	ARPROTMx	As original transaction
	ARLOCKMx	{00}
	ARADDRMx[2:0]	3'b000
	ARLENMx	4'b0011
	ARSIZEx	2'b11 (64-bit)
	ARBURSTMx	2'b10 (WRAP)

Table 2-6 Exported master ports AXI control signals (continued)

Access type	Master control signals buses	Value (Verilog)
Read or write from store buffer, including write-through	ARCACHEM_x/AWCACHEM_x	Depends on cacheable attributes of the store buffer slot: {0001} for Device {0011} for Normal Memory Non-Cacheable {0110} for Write-Through Non Write-Allocate {1110} for Write-Through Write-Allocate {0111} for Write-Back Non Write-Allocate {1111} for Write-Back Write-Allocate
	ARPROTM_x/AWPROTM_x	{0, SECURITY, 1}
	ARLOCKM_x/AWLOCKM_x	{00}
	ARADDRM_x[2:0]	3'b000
	ARLENM_x	4'b0011
	ARSIZEM_x	2'b11 (64-bit)
	ARBURSTM_x	2'b01 (INCR)
	AWADDRM_x[2:0]	<ul style="list-style-type: none"> 3'b000 if Normal Memory depends on original transaction if Device Memory.
	AWLENM_x	between 4'b0000 and 4'b0011
	AWSIZEM_x	<ul style="list-style-type: none"> 2'b11 (64-bit) if Normal Memory depends on original transaction if Device Memory.
	AWBURSTM_x	2'b01 (INCR)
Evictions	AWCACHEM_x	{1111}
	AWPROTM_x	{0, SECURITY, 1}
	AWLOCKM_x	{00}
	AWADDRM_x[3:0]	4'b0000
	AWLENM_x	4'b0011
	AWSIZEM_x	2'b11 (64-bit)
	AWBURSTM_x	2'b01 (INCR)

Note

SECURITY denotes the value of the NS attribute stored with the data.

2.2.5 AXI locked and exclusive accesses

The AXI locked and exclusive accesses are described in:

- *AXI locked transfers*
- *AXI exclusive accesses* on page 2-12.

AXI locked transfers

In the case of a noncacheable transfer, the access is forwarded to L3 memory through the master ports and is marked as locked.

In the case of cacheable transfers, a cache lookup is always performed. In the case of a cache miss, a linefill (non-locked) is requested on the master side. Write accesses always cause non-locked writes on the master side.

When a slave is performing a locked sequence, cacheable or not, the other slave is stopped from accepting more transfers. A locked transaction is stalled until all buffers are empty, including the store buffer.

The processor must ensure that there is only one outstanding transaction across the read and write channels during a locked sequence.

If multiple locked transfers come in at the same time, they are permitted to proceed in a certain priority. The priority for locked transfers is that **S0** takes priority over **S1**.

Note

A locked sequence must consist of solely noncacheable or cacheable transactions. A locked sequence cannot contain a mix of cacheable and noncacheable transactions. The cache controller does not support a locked sequence starting with one locked read and one locked write at the same time on the same slave port.

AXI exclusive accesses

The control signals for the read and write portions of the exclusive access must be identical.

The cache controller supports cacheable and non-cacheable exclusive accesses but does not provide an exclusive monitor. The system integrator must implement external exclusive monitors as follows, so that the EXOKAY response can be returned:

- for cacheable exclusive accesses, implement one or more external exclusive monitors on the slave side of the cache controller
- for non-cacheable exclusive accesses, implement one or more external exclusive monitors on the master side of the cache controller.

The monitor on the slave side must be aware of the cache controller internal status, such as the shared override bit, to determine which accesses are cacheable and which are not.

Note

All exclusive accesses to the cache controller configuration registers return a SLVERR response.

2.3 Cache operation

Table 2-7 to Table 2-15 on page 2-16 show the general behavior of the cache controller depending on ARMv6 and ARMv7 transactions.

Table 2-7 shows the general behavior of the cache controller for Noncacheable and nonbufferable AXI transactions.

Table 2-7 Noncacheable and nonbufferable AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Strongly ordered	Read: Not cached in L2, results in L3 access. Write: Not buffered, results in L3 access.

Table 2-8 shows the general behavior of the cache controller for Bufferable only AXI transactions.

Table 2-8 Bufferable only AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Device	Read: Not cached in L2, results in L3 access. Write: Put in store buffer, not merged, immediately drained to L3. ^a

- a. Not all types of Device writes go to the store buffer. For example, the L2CC (PL310) treats a Device write that crosses a cache line boundary as a Strongly Ordered access.

Table 2-9 shows the general behavior of the cache controller for Chacheable but do not allocate AXI transactions, and Cacheable and bufferable but do not allocate AXI transactions.

Table 2-9 Cacheable but do not allocate AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Outer non cacheable	Read: Not cached in L2, results in L3 access. Write: Put in store buffer, write to L3 when store buffer is drained.

Table 2-10 shows the general behavior of the cache controller for Cacheable write-through, allocate on read AXI transactions.

Table 2-10 Cacheable write-through, allocate on read AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Outer write-through, no write allocate	Read hit: Read from L2. Read miss: Linefill to L2. Write hit: Put in store buffer, write to L2 and L3 when store buffer is drained. Write miss: Put in store buffer, write to L3 when store buffer is drained.

Table 2-11 shows the general behavior of the cache controller for Cacheable write-back, allocate on read AXI transactions.

Table 2-11 Cacheable write-back, allocate on read AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Outer write-back, no write allocate	Read hit: Read from L2. Read miss: Linefill to L2. Write hit: Put in store buffer, write to the L2 when store buffer is drained, mark line as dirty. Write miss: Put in store buffer, write to L3 when store buffer is drained.

Table 2-12 shows the general behavior of the cache controller for Cacheable write-through, allocate on write AXI transactions.

Table 2-12 Cacheable write-through, allocate on write AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
	Read hit: Read from L2. Read miss: Not cached in L2, causes L3 access. Write hit: Put in store buffer, write to L2 and L3 when store buffer is drained. Write miss: <ul style="list-style-type: none">• Put in store buffer.• When buffer has to be drained, check whether it is full. If it is not full then request word or line to L3 before allocating the buffer to the L2.• Allocation to L2.• Write to L3.

Table 2-13 shows the general behavior of the cache controller for Cacheable write-back, allocate on write AXI transactions.

Table 2-13 Cacheable write-back, allocate on write AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
	Read hit: Read from L2. Read miss: Not cached in L2, causes L3 access. Write hit: Put in store buffer, write to the L2 when store buffer is drained, mark line as dirty. Write miss: <ul style="list-style-type: none">• Put in store buffer.• When buffer has to be drained, check whether it is full. If it is not full then request word or line to L3 before allocating the buffer to the L2.• Allocation to L2.

Table 2-14 shows the general behavior of the cache controller for Cacheable write-through, allocate on read and write AXI transactions.

Table 2-14 Cacheable write-through, allocate on read and write AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Outer write-through, allocate on both reads and writes	Read hit: Read from L2. Read miss: Linefill to L2. Write hit: Put in store buffer, write to L2 and L3 when store buffer is drained. Write miss: <ul style="list-style-type: none">• Put in store buffer.• When buffer has to be drained, check whether it is full. If it is not full then request word or line to L3 before allocating the buffer to the L2.• Allocation to L2.• Write to L3.

Table 2-15 shows the general behavior of the cache controller for Cacheable write-back, allocate on read and write AXI transactions.

Table 2-15 Cacheable write-back, allocate on read and write AXI transactions

ARMv6 and ARMv7 memory type attribute	Cache controller behavior
Outer write-back, write allocate	Read hit: Read from L2. Read miss: Linefill to L2. Write hit: Put in store buffer, write to L2 when store buffer is drained, mark line as dirty. Write miss: <ul style="list-style-type: none">• Put in store buffer.• When buffer has to be drained, check whether it is full. If it is not full then request word or line to L3 before allocating the buffer to the L2.• Allocation to L2.

Note

You can modify the default behavior described in Table 2-7 on page 2-13 to Table 2-15 on page 2-16 using parameters, such as shared attribute, force write allocate, and exclusive cache configuration.

Other behaviors are described in:

- *Shared attribute* on page 2-19
- *Force write allocate* on page 2-20
- *Exclusive cache configuration* on page 2-21.

2.3.1 Cache attributes

Table 2-16 describes the **AWCACHE[3:0]** and **ARCACHE[3:0]** signals as described in the AMBA AXI Protocol Specification, and the ARMv6 and ARMv7 equivalent meaning. Table 2-16 does not show AXI locked and exclusive accesses.

Table 2-16 AWCACHE and ARCACHE definitions

AWCACHE/ARCACHE				AXI meaning	ARMv6 and ARMv7 equivalent
WA	RA	C	B		
0	0	0	0	Noncacheable, nonbufferable	Strongly ordered
0	0	0	1	Bufferable only	Device
0	0	1	0	Cacheable but do not allocate	Outer noncacheable
0	0	1	1	Cacheable and bufferable, do not allocate	Outer noncacheable
0	1	1	0	Cacheable write-through, allocate on read	Outer write-through, no allocate on write
0	1	1	1	Cacheable write-back, allocate on read	Outer write-back, no allocate on write
1	0	1	0	Cacheable write-through, allocate on write	-
1	0	1	1	Cacheable write-back, allocate on write	-
1	1	1	0	Cacheable write-through, allocate on both read and write	-
1	1	1	1	Cacheable write-back, allocate on both read and write	Outer write-back, write allocate

Note

- The shared attribute **AW/RUSERSx[0]** is not described in this table. Its behavior is explained in *Cache operation* on page 2-13, under *Shared attribute*.
 - The cache controller supports all AXI cache attributes, even if the processor does not use all of them.
 - If the cache controller receives cacheable fixed transactions, **AWBURST** or **ARBURSTSx = 00**, the results are unpredictable.
-

2.3.2 Shared attribute

The **ARUSERSx[0]** and **AWUSERSx[0]** signals affect transactions. Typically, these signals are driven by ARM processors and reflect the shared attribute as defined in the ARM v6 and v7 architecture.

Shared only applies to Normal Memory outer non-cacheable transactions, where **ARCACHESx** or **AWCACHESx** = 0010 or 0011. For other values of **ARCACHESx** and **AWCACHESx**, the shared attribute is ignored.

The default behavior of the cache controller with respect to the shared attribute is to transform Normal Memory Non-Cacheable transactions into:

- cacheable no allocate for reads
- write through no write allocate for writes.

You can change this default shared behavior by setting the **Shared Attribute Invalidate Enable** bit in the Auxiliary Control Register (bit 13). When you set this bit, writes targeting a full cache line (for example 4x64 bit bursts with all strobes active) and hitting in the L2 cache invalidate the corresponding cache line and are forwarded to L3. Other cases are identical to the default shared behavior.

Note

The **Shared Attribute Invalidate Enable** bit can cause the invalidation of L2 cache lines even if they are dirty. So you must only enable this bit in systems supporting this behavior. When you set this bit in such systems, all non-cacheable writes marked as shared must be 4x64-bit bursts targeting a full cache line. Otherwise, the behavior might be unpredictable.

Both of these shared behaviors are disabled if you set the **Shared Attribute Override Enable** bit in the Auxiliary Control Register (bit [22]).

Note

- Dynamically changing the **Shared Attribute Override Enable** bit without flushing the cache could cause a hazard where incorrect data could be evicted causing more recent data in L3 to be overwritten.
 - The behavior of the L2CC (PL310) with respect to the shared attribute is different from the L220 Level 2 cache controller. Take care when moving from an L220 based system to a system implementing L2CC (PL310), because the shared attribute and the **Shared Attribute Override Enable** bit affects the point of coherency of such systems.
-

2.3.3 Force write allocate

The default setting for the Force write allocate bits in the Auxiliary Control Register is 00, and this configures the cache controller to use the received **AWCACHE** or **ARCACHE** attributes. Additional reference data on the general behavior can be found in the set of tables from Table 2-7 on page 2-13 to Table 2-16 on page 2-17.

If you set the Force write allocate bits in the Auxiliary Control Register to 01, this causes the WA bit to be always set to 0. No write allocation is performed.

If you set the Force write allocate bits in the Auxiliary Control Register to 10, this causes cacheable write misses to be allocated in the cache.

Note

- The **AWUSERSx[0]** signal takes priority over the force write allocate settings, that is, if **AWUSERSx[0]** is set it causes the transaction to be no write-allocate.
 - The Force Write Allocate configurations do not change the **ARCACHE** or **AWCACHE** attributes present on the master ports to L3.
 - The Force Write allocate feature has priority over the exclusive cache configuration behavior described in *Exclusive cache configuration* on page 2-21.
-

2.3.4 Exclusive cache configuration

Setting the exclusive cache configuration bit in the Auxiliary Control Register to 1 configures the L2 cache to behave as an exclusive cache relative to the L1 cache. The exclusive cache mechanism only applies to the outer write-back inner write-back data transactions received by the cache controller slave ports, that is,

ARCACHESx/AWCACHESx = ARUSER[4:1]/AWUSER[4:1] = 1011 or 0111 or 1111.

Reads

For reads, the behavior is as follows:

- In case of a hit, the line is marked as non-valid, that is, the Tag RAM valid bit is reset, and the dirty bit is unchanged. If the dirty bit is set, future accesses can still hit in this cache line but the line is part of the preferred choice for future evictions.
- In case of a miss, the line is not allocated into the L2 cache.

Writes

For writes, the behavior depends on the value of **AWUSERSx[9:8]**. **AWUSERSx[8]** indicates that the write transaction is an eviction from the L1 memory system. **AWUSERSx[9]** indicates if this eviction is clean.

- In case of a hit, the line is marked dirty unless **AWUSERSx[9:8] = 11**. In this case, the dirty bit is unchanged.
- In case of a miss, if **AWUSERSx[8]** is HIGH, the cache line is allocated and its dirty status depends on the value of **AWUSERSx[9]**. If **AWUSERSx[8]** is LOW, the cache line is allocated only if it is write allocate.

2.3.5 TrustZone support in the cache controller

Some aspects of TrustZone support for the cache controller are:

- The cache controller attaches an NS bit to all data stored in the L2 cache and in internal buffers. A Non Secure, resp. Secure, transaction cannot access Secure, resp. Non Secure, data. Therefore the controller treats Secure and Non-Secure data as being part of two different memory spaces.
- The controller treats as a miss a Non Secure, resp. Secure, access to data in the L2 cache that is marked Secure, resp. Non Secure.

For a read transfer the cache controller:

1. Sends a linefill command to L3 memory.

2. Propagates any security errors from L3 to the processor.
3. Does not allocate the line in L2.

The type of the L3 generated security errors is specific to the L3 cache and therefore outside the scope of this document.

- You can only write to the L2 Control Register with an access tagged as secure, to enable or disable the L2 cache.
- You can only write to the Auxiliary Control Register with an access tagged as secure.
- NS maintenance operations do not clean or invalidate secure data.
- Bit [26] in the Auxiliary Control Register is for NS Lockdown enable. You can only modify it with secure accesses. Use this bit to determine whether NS accesses can modify a lockdown register.

2.3.6 Cache Lockdown

You can use these lockdown mechanisms in the L2CC (PL310):

- lockdown by line (optional)
- lockdown by way
- lockdown by master (optional).

You can use the lockdown by line and the lockdown by way at the same time. You can also use the lockdown by line and the lockdown by master at the same time. But lockdown by master and lockdown by way are exclusive because lockdown by way is a subset of the lockdown by master.

The Lockdown by master feature is optional. You can only implement it if you define the parameter `p1310_LOCKDOWN_BY_MASTER`, see *Cache configurability* on page 2-2.

Lockdown by line

This feature is optional. You can only implement it if you define the parameter `p1310_LOCKDOWN_BY_LINE`, see *Cache configurability* on page 2-2.

When enabled during a period of time, all newly allocated cache lines get marked as locked. The controller then considers them as locked and does not naturally evict them. You enable it by setting bit [0] of the Lockdown by Line Enable Register. see Table 3-18 on page 3-35. The optional bit [21] of the Tag RAM shows the locked status of each cache line.

Note

An example of when you might enable the lockdown by line feature is during the time when you load a critical piece of software code into the L2 cache.

The Unlock All Lines background operation enables you to unlock all lines marked as locked by the Lockdown by Line mechanism. You can check the status of this operation by reading the Unlock All Lines register, see Table 3-19 on page 3-35.

Lockdown by way

The 32-bit ADDR cache address consists of the following fields:

- < TAG > < INDEX > < WORD > < BYTE >.

When a cache lookup occurs, the Index defines where to look in the cache ways. The number of ways defines the number of locations with the same Index. This is called a Set. Therefore a 16-way set associative cache has 16 locations where an address with INDEX (A) can exist.

The Lockdown format C, as the *ARM Architecture Reference Manual* describes, provides a method to restrict the replacement algorithm used for allocations of cache lines within a Set.

This method enables you to:

- fetch code or load data into the L2 cache
- protect it from being evicted.

You can also use this method to reduce cache pollution.

You can use two registers to control this mechanism, see Table 3-20 on page 3-35 and Table 3-21 on page 3-35.

Lockdown by master

This feature enables you to reserve ways of the L2 cache to specific L1 masters that the ID on the **AR/WUSERSx[7:5]** (x=0,1) signals identify.

You can use sixteen registers for controlling this mechanism. See the tables from Table 3-20 on page 3-35 to Table 3-35 on page 3-38.

System including four Cortex-A9 MPCore processors and L2CC (PL310) on page 2-24 and *System including one Cortex-A9 MPCore processor with ACP and L2CC (PL310)* on page 2-25 describe two examples of a system that use the lockdown by master. They show you how to drive **AR/WUSERSx[7:5]** signals and program the relevant registers.

System including four Cortex-A9 MPCore processors and L2CC (PL310)

Figure 2-3 shows a system where a Cortex-A9 cluster with four CPUs drives the L2CC (PL310).

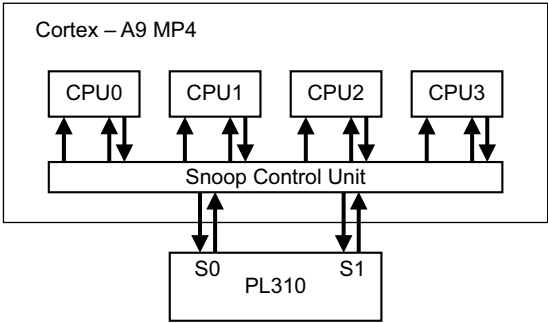


Figure 2-3 Driven by cortex a9 cluster with 4 cpus

If you implement the L2CC (PL310) with a 16-way associativity, the lockdown by master feature enables you to reserve four ways for each CPU. If you want to reserve four ways for each CPU, they are for allocations only. All CPUs have access to all of the ways for lookups and hits.

For this system you can drive the **AR/WUSERSx[7:5]** signals as follows:
AR/WUSERSx[7:5] = { 1'b0, AR/WIDMx[1:0]}.

Note

AR/WIDMx[1:0] are the AXI ID bits that the Cortex-A9 cluster drives, see the *Cortex-A9 TRM*.

Table 2-17 shows how you can program the lockdown registers.

Table 2-17 MP4 system lockdown register definitions

Register	Offset	Value
Data Lockdown 0 ^a	0x900	0x0000FFF0
Instruction Lockdown 0	0x904	0x0000FFF0
Data Lockdown 1	0x908	0x0000FF0F
Instruction Lockdown 1	0x90C	0x0000FF0F
Data Lockdown 2	0x910	0x0000F0FF

Table 2-17 MP4 system lockdown register definitions (continued)

Register	Offset	Value
Instruction Lockdown 2	0x914	0x0000F0FF
Data Lockdown 3	0x918	0x00000FFF
Instruction Lockdown 3 ^b	0x91C	0x00000FFF
All other Lockdown registers	0x920 - 0x93C	default

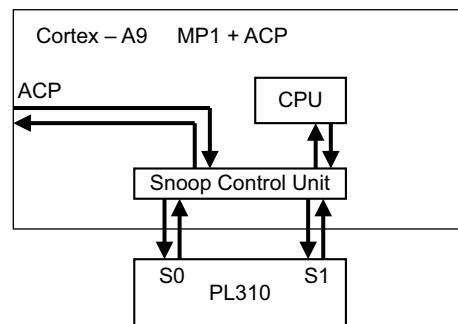
- a. Use when AR/WUSERSx[7:5] = 000 (transactions from CPU0), and AR/WPROTSx[2] = 0.
- b. Use when AR/WUSERSx[7:5] = 011 (transactions from CPU3), and AR/WPROTSx[2] = 1.

This configuration reserves:

- ways [0:3] for CPU0
- ways [4:7] for CPU1
- ways [8:11] for CPU2
- ways [12:15] for CPU3.

System including one Cortex-A9 MPCore processor with ACP and L2CC (PL310)

Figure 2-4 shows a system where the Cortex-A9 cluster has only one CPU and the *Accelerator Coherence Port* (ACP) drives the L2CC (PL310).

**Figure 2-4 Driven by cortex a9 cluster with 1 CPU and ACP**

If you implement the L2CC (PL310) with 8-way associativity, the lockdown by master feature enables you to reserve four ways for the CPU and the other four ways for the master to drive the ACP.

For this system you can drive the AR/WUSERSx[7:5] signals as follows:

- AR/WUSERSx[7:5] = {2'b00, AR/WIDMx[2]}.

———— **Note** —————

AR/WIDMx are the AXI ID bits that the Cortex-A9 cluster drives, see the *Cortex-A9 TRM*.

Table 2-18 shows how you can program the lockdown registers.

Table 2-18 MPI plus APC system lockdown register definitions.

Register	Offset	Value
Data Lockdown 0 ^a	0x900	0x000000F0
Instruction Lockdown 0	0x904	0x000000F0
Data Lockdown 1	0x908	0x0000000F
Instruction Lockdown 1 ^b	0x90C	0x0000000F
All other Lockdown registers	0x910 - 0x93C	default

- a. use when AR/WUSERSx[7:5] = 000 (transactions from CPU), and AR/WPROTSx[2] = 0.
- b. use when AR/WUSERSx[7:5] = 001 (transactions from ACP), and AR/WPROTSx[2] = 1.

When you have programmed the registers, and the cache controller drives AR/WUSERSx[7:5] as *Lockdown by way* on page 2-23 shows, the cache controller reserves:

- ways [3:0] for the Cortex-A9 CPU
- ways [7:4] for the ACP.

2.4 RAM interfaces

This section describes:

- *RAM organization*
- *RAM clocking and latencies* on page 2-34
- *MBIST support* on page 2-37.

2.4.1 RAM organization

This section describes:

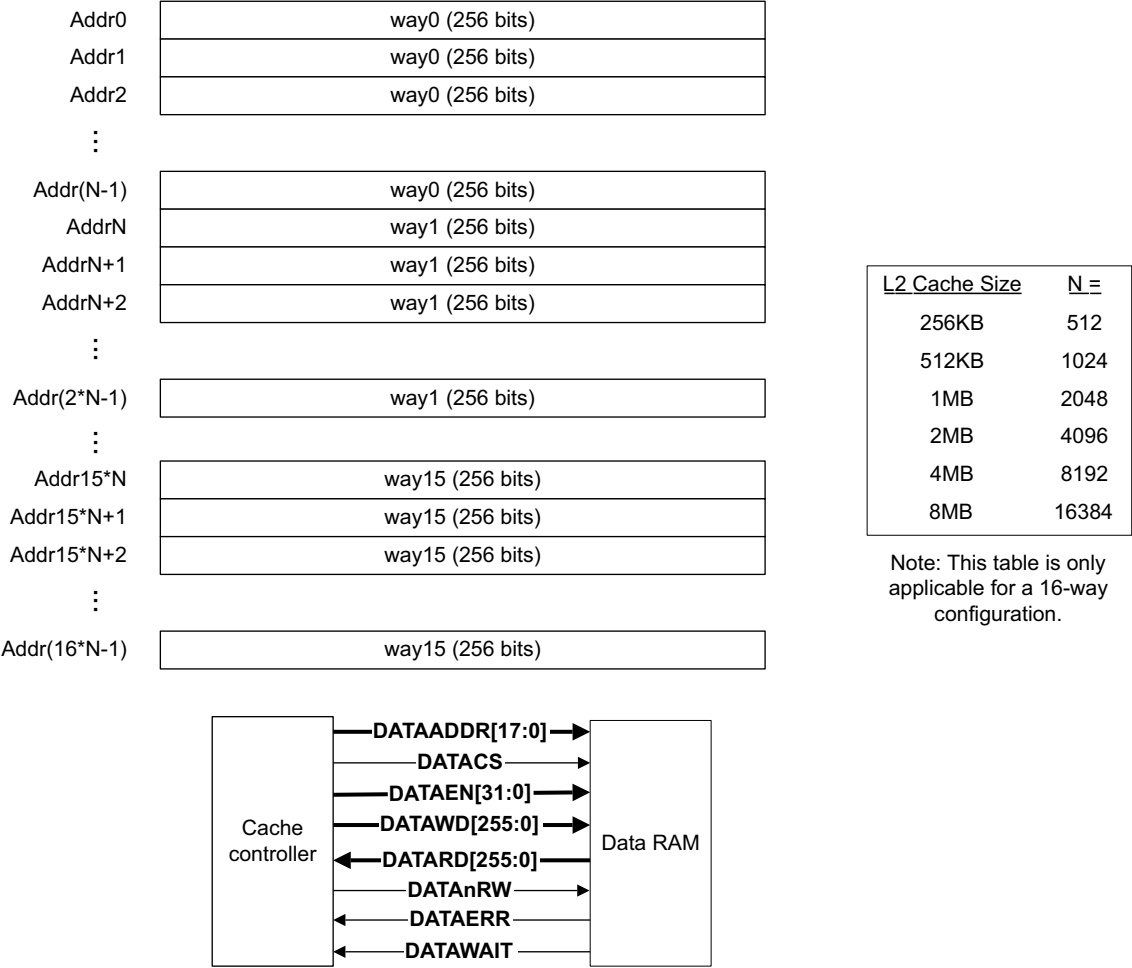
- *Data RAM*
- *Tag RAM* on page 2-29
- *Parity RAM* on page 2-30
- *RAM bus usage versus cache associativity and way size* on page 2-31.

See the *ARM PL310 Cache Controller Implementation Guide* for more information about RAM organization.

Data RAM

The data RAM shown in Figure 2-5 on page 2-28 is organized as n -way 256-bit wide contiguous memories, where n is 8 or 16. It supports the following accesses:

- 8 word data reads
- 256 bit data writes with byte enables controls
- 8 word data writes for line allocations.



Cache controller

DATAADDR[17:0]

DATAACS

DATAEN[31:0]

DATAWD[255:0]

DATARD[255:0]

DATANRW

DATAERR

DATAWAIT

Data RAM

Figure 2-5 Data RAM organization for 16 ways

The Data RAM interface signal **DATAWAIT** provides support for *Error Correction Code* (ECC) through an external block. When the Data RAM latency is reached, the cache controller still waits for the data or error if the wait signal is asserted. The data or error is then sampled when the wait signal is deasserted.

Tag RAM

Figure 2-6 on page 2-30 shows an example of tag RAM. There is one tag RAM for each way of the L2 cache. A tag RAM is organized as:

- A maximum of 18 bits for address tag depending on way size. See *RAM bus usage versus cache associativity and way size* on page 2-31 for more information.
- 1 bit for security information.
- 1 bit for valid information.
- 1 bit for dirty information.
- 1 optional bit for lock information.
- 1 optional bit for parity.

The NS bit takes the value of 1 for NS data, and 0 for secure data.

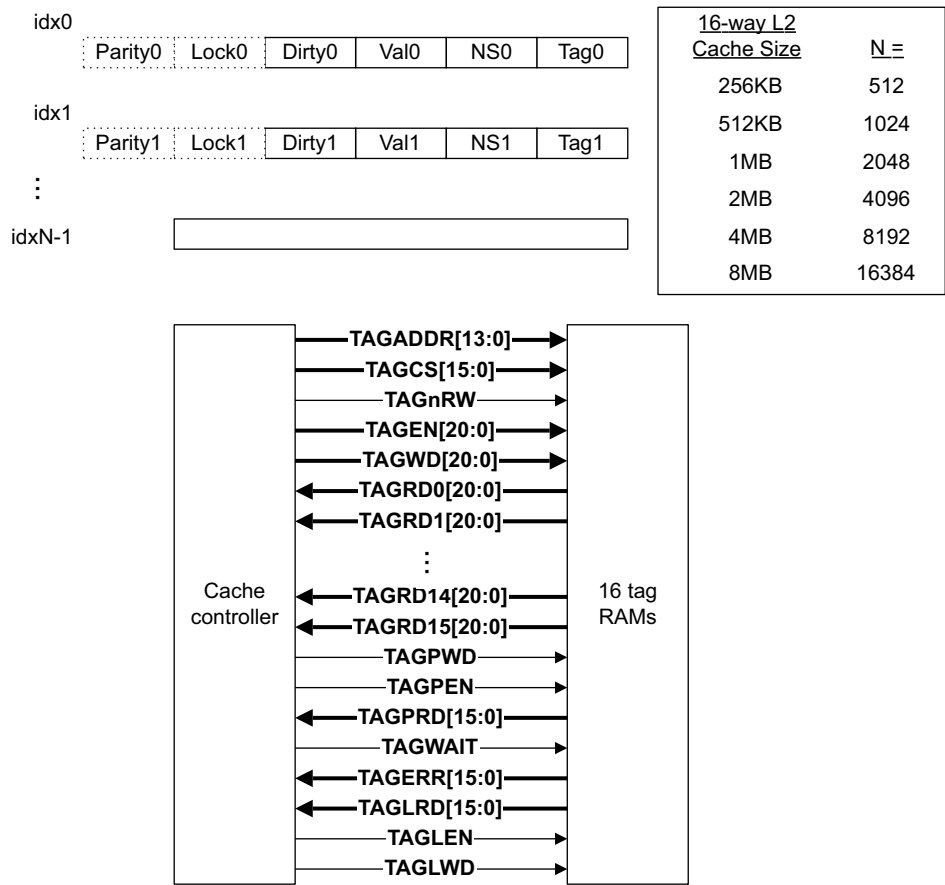


Figure 2-6 Tag RAM organization for a 16-way 256KB L2 cache, with parity, with lockdown by line

A **TAGWAIT** signal is added to the tag RAM interface to provide support for ECC through an external block. When the tag RAM latency is reached, the cache controller still waits for the tags or error if the wait signal is asserted. The tags or error are then sampled when the wait signal is deasserted.

Parity RAM

Optional parity RAM is described in:

- Tag parity RAM on page 2-31
- Data parity RAM on page 2-31.

Tag parity RAM

Because there is only one tag parity bit per way, the simplest implementation of tag parity RAM is to have tag RAMs one bit wider and connect the respective **TAGPRD** on additional read data bus bits and **TAGPWD** signals on additional write data buses. Tag parity RAMs can be split from tag RAM. Additional logic is required on tag parity RAM control signals.

Data parity RAM

Figure 2-7 shows the required RAM organization for data parity RAM.

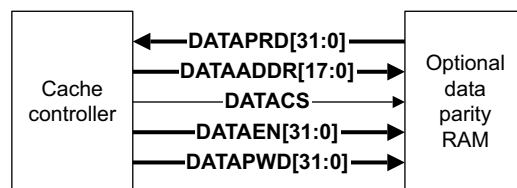


Figure 2-7 Data parity RAM organization

Data parity RAM is always 32 bits wide, and connection is the same as for data RAM described in *Data RAM* on page 2-27.

RAM bus usage versus cache associativity and way size

This section describes:

- Data RAM usage
- Tag RAM usage.

Data RAM usage

Figure 2-8 shows the DATAADDR bus format:

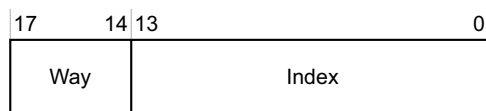


Figure 2-8 Data RAM address bus format for 16 ways

Note

For 8 ways, the Way field is [16:14].

Bits [13:0] usage depends on the way size:

- 16KB way size:
 - Bits [13:9] left unconnected.
 - Bits [8:0] are the LSB of the RAM address bus.
- 32KB way size:
 - Bits [13:10] are left unconnected.
 - Bits [9:0] are the *Least Significant Bits* (LSB) of the RAM address bus.
- 64KB way size:
 - Bits [13:11] are left unconnected.
 - Bits [10:0] are the LSB of the RAM address bus.
- 128KB way size:
 - Bit [13:12] is left unconnected.
 - Bits [11:0] are the LSB of the RAM address bus.
- 256KB way size:
 - Bit [13] is left unconnected.
 - Bits [12:0] are the LSB of the RAM address bus.
- 512KB way size:
 - All [13:0] bits are used.

All bits of the data buses, **DATARD** and **DATAWD**, are always connected.

Tag RAM usage

TAGADDR and **TAGRDn/TAGWD** bus usage depends on the way size:

- 16KB way size:
 - Bits [13:9] of **TAGADDR** are left unconnected.
 - Bits [8:0] are the RAM address bus. All bits of **TAGRDn** and **TAGWD** are used for data buses.
- 32KB way size:
 - Bits [13:10] of **TAGADDR** are left unconnected.
 - Bits [9:0] are the RAM address bus.
 - Bits [20:1] of **TAGRDn** and **TAGWD** are used for data buses.
 - Bit [0] of **TAGRDn** must be tied LOW.

- 64KB way size:
 - Bits [13:11] of **TAGADDR** are left unconnected.
 - Bits [10:0] are the RAM address bus.
 - Bits [20:2] of **TAGRDn** and **TAGWD** are used for data buses.
 - Bits [1:0] of **TAGRDn** must be tied LOW.
- 128KB way size:
 - Bits [13:12] of **TAGADDR** are left unconnected.
 - Bits [11:0] are the RAM address bus.
 - Bits [20:3] of **TAGRDn** and **TAGWD** are used for data buses.
 - Bits [2:0] of **TAGRDn** must be tied LOW.
- 256KB way size:
 - Bit [13] of **TAGADDR** is left unconnected.
 - Bits [20:4] of **TAGRDn** and **TAGWD** are used for data buses.
 - Bits [3:0] of **TAGRDn** must be tied LOW.
- 512KB way size:
 - All bits [13:0] are the RAM address bus.
 - Bits [20:5] of **TAGRDn** and **TAGWD** are used for data buses.
 - Bits [4:0] of **TAGRDn** must be tied LOW.

Note

The Tag RAM width and the connections of TAGADDR depend on the way size that you choose. It is possible to change the way size by writing to the Auxiliary Control Register. An example of a typical usage model is to implement a certain way size but then make it appear smaller than the actual implementation. To support this you must implement the Tag RAM with specific requirements as follows:

- connect TAGADDR to support the biggest way_size
 - TAGRDn and TAGWD (that correspond to the Tag RAM width) must be connected to support the smallest way size.
-

2.4.2 RAM clocking and latencies

This section describes:

- *RAM clocking*
- *RAM latencies* on page 2-36.

———— **Note** ————

The text and figures in this section apply to both Tag RAM and Data RAM.

RAM clocking

Clock enables can be used if the RAMs are run at a slower frequency than the cache controller logic. Only integer ratios are supported. With this scheme, the Tag RAM and the Data RAM can run at different frequencies. Table 2-19 shows the RAM clock enables and their functions.

Table 2-19 RAM clock enables

Signal	Function
TAGCLKEN	Clock enable input that enables the Tag RAM interface in the cache controller to communicate with the Tag RAM clocked at a slower frequency.
TAGCLKOUTEN	Clock enable output used to gate the clock to the Tag RAM to save power. Use this signal when you run the Tag RAM at a slower frequency than the cache controller logic.
TAGCLKOUT	Gated version of CLK that is only enabled when the Tag RAM is accessed. Use this clock if you run the Tag RAM and the cache controller logic at the same frequency.
DATACLKEN	Clock enable input that enables the Data RAM interface in the cache controller to communicate with the Data RAM clocked at a slower frequency.
DATACLKOUTEN	Clock enable output used to gate the clock to the Data RAM to save power. Use this signal when you run the Data RAM at a slower frequency than the cache controller logic.
DATACLKOUT	Gated version of CLK that is only enabled when the Data RAM is accessed. Use this clock you run the Data RAM and the cache controller logic at the same frequency.

Figure 2-9 on page 2-35 shows an example of the Tag RAM running at half the frequency compared to the cache controller.

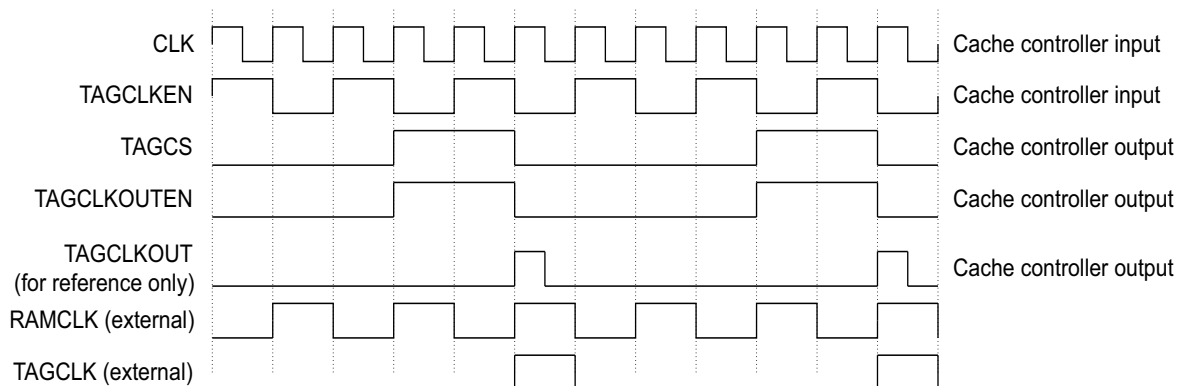


Figure 2-9 Tag RAM running at slower frequency

Figure 2-10 shows how the different clock gates used in the Tag RAM clocking are expected to be implemented.

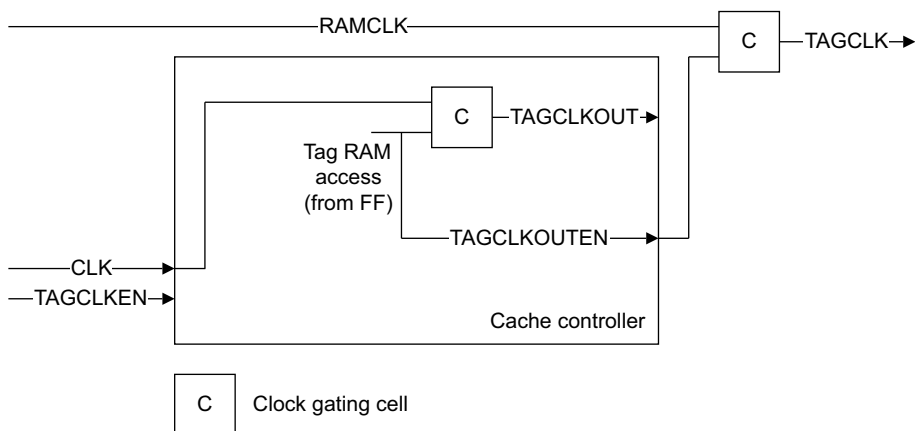


Figure 2-10 Tag RAM clock gating

Note

The implementer of the ram array must also be the one that implements the clock gating cell that outputs **TAGCLK** in Figure 2-10.

RAM latencies

Programmable RAM latencies enable the cache controller to manage RAMs requiring several clock cycles for dealing with accesses. For each RAM, there are three programmable latencies:

- setup
- read access
- write access.

See *Tag and Data RAM Latency Control Registers* on page 3-15.

Setup latency is the number of cycles that the RAM control signals remain valid prior to the RAM clock edge. Figure 2-11 shows a timing diagram where the Tag RAM setup latency has been programmed with the value 0x1.

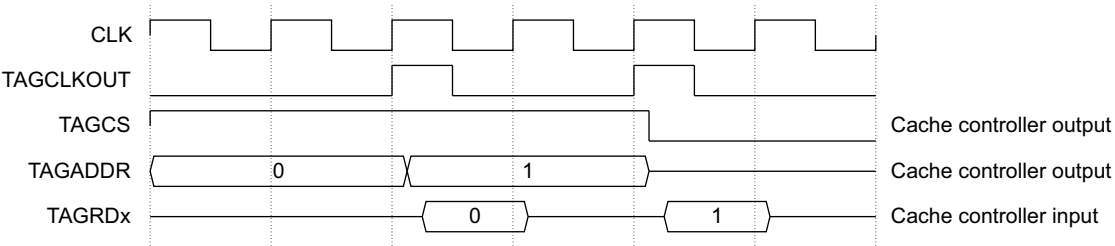


Figure 2-11 Tag RAM setup latency

Read access latency is the number of cycles taken by the read data to become valid after the RAM clock edge. Figure 2-12 shows a timing diagram where the Tag RAM read access latency has been programmed with the value 0x1.

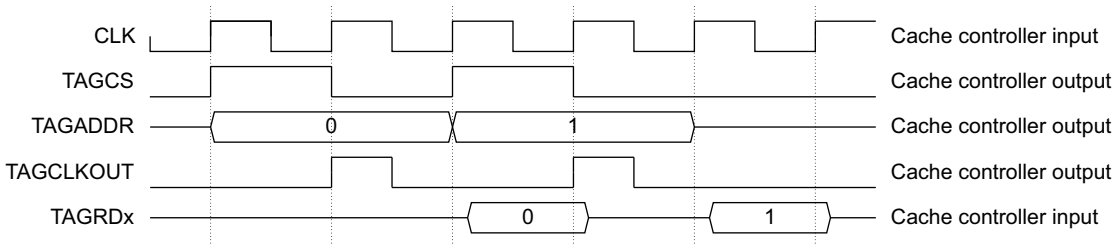


Figure 2-12 Tag RAM read access latency

Write access latency is the minimum number of cycles between a RAM clock edge for a write access and the next RAM clock edge corresponding to another access (read or write). Figure 2-13 on page 2-37 shows a timing diagram where the Tag RAM write access latency has been programmed with the value 0x1.

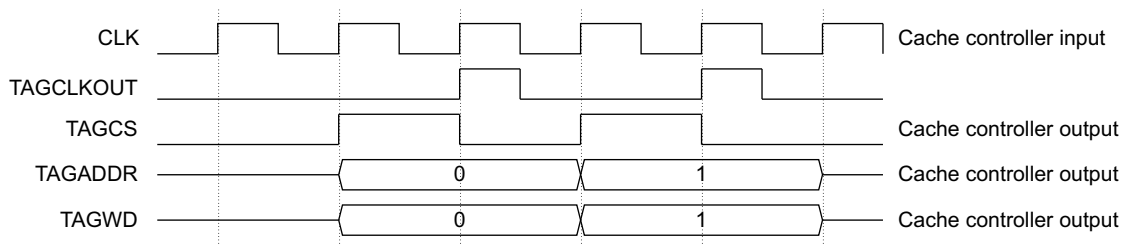


Figure 2-13 Tag RAM write access latency

Note

The programmed RAM latency values refer to the actual clock driving the RAMs, that is, the slow clock if the RAMs are clocked slower than the cache controller.

2.4.3 MBIST support

MBIST is used for testing the RAMs connected to the cache controller. ARM can supply an MBIST controller as a separate block, or you can design your own MBIST controller. Only one RAM can be accessed by the MBIST port at a time.

The data RAM is 256 bits wide, and the size of the **MBISTDIN** and **MBISTDOUT** buses on the cache controller is 64 bits, so four reads and four writes are required for each index of the data RAM. The cache controller handles this by using two of the **MBISTADDR** address pins as a double word select for each index of the data RAM.

The MBIST controller must be able to account for the different latencies of the RAMs. Data read, data write, and tag read or write accesses can all be programmed with different access latencies.

If present, the tag parity bit is tested at the same time as tag RAMs. Parity bits are considered as an extra bit on tag data bus.

Figure 2-14 on page 2-38 shows the cache controller MBIST interface.

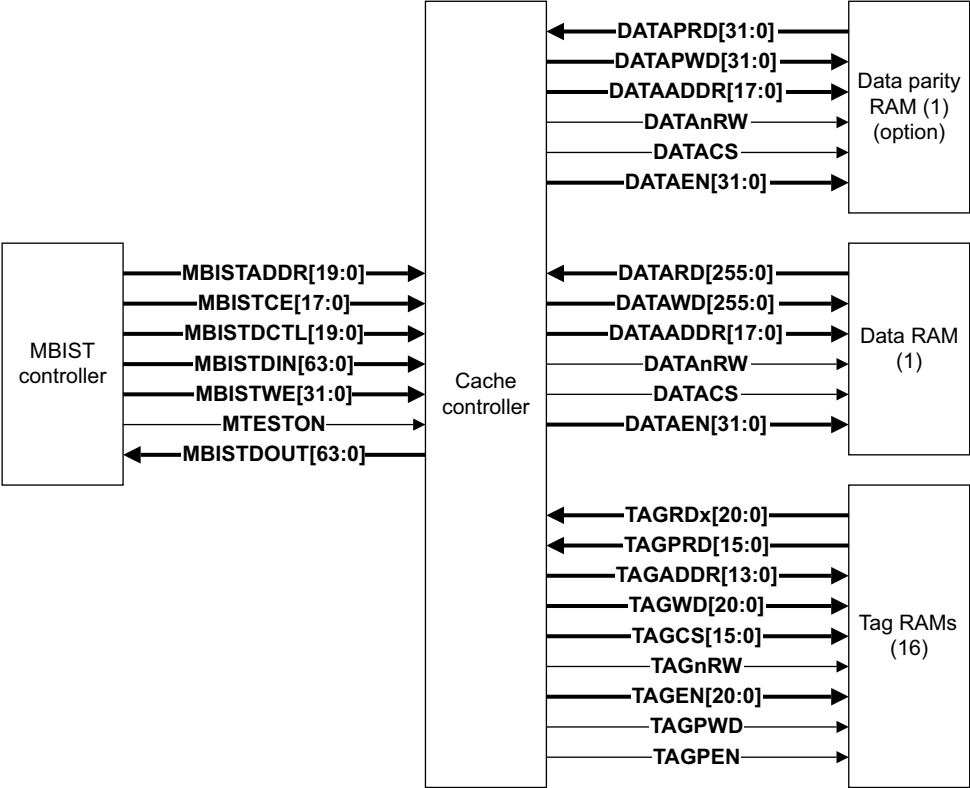


Figure 2-14 MBIST interface for 16-way implementation, with parity, without lockdown by line

Note

MBIST is a secure feature. The signals are available at the top level of the design for test, but must not be bonded out in production.

2.5 Implementation details

This section describes:

- *Disabled operation*
- *Store buffer operation*
- *Cache configurability* on page 2-2
- *Hazards* on page 2-40
- *External error support for L3 memory* on page 2-42
- *Cache event monitoring* on page 2-44
- *Cache interrupts outputs* on page 2-45
- *Parity and RAM error support* on page 2-45.

2.5.1 Disabled operation

When the cache controller block is present but not enabled, transactions are passed through to the L3 memory system on the cache controller master ports. The address latency introduced by the disabled cache controller is one cycle in the slave ports plus one cycle in the master ports.

2.5.2 Store buffer operation

Two buffered write accesses to the same address and the same security bit, cause the first one to be overridden if the controller does not drain the store buffer after the first access.

The store buffer has merging capabilities so it merges successive writes, to a same line address, into the same buffer slot. This means that the controller does not drain the slots as soon as they contain data but waits for potential other accesses that target the same cache line. The store buffer draining policy is:

- Store buffer slot is immediately drained if targeting device memory area.
- Store buffer slots are drained as soon as they are full.
- Store buffer is drained at each strongly ordered read occurrence in slave port(s).
- Store buffer is drained at each strongly ordered write occurrence in slave port(s).
- If the three slots of the store buffer contain data, the least recently accessed is drained.
- If a hazard is detected with one store buffer slot, it is drained to resolve the hazard.
- Store buffer slots are drained when a locked transaction is received by one slave port.

- Store buffer slots are drained when a transaction targeting the configuration registers is received by one slave port.

Merging condition is based on address and security attribute, **AWPROTS0[1]** or **AWPROTS1[1]**. Merging takes place only when data is in the store buffer and it is not draining.

The store buffer has three data slots that each contain a 256-bit data line. Each data slot contains a byte-valid field that enables the control logic to determine the data line fill level. Each data slot is attached to two address slots with address and security information.

When a write-allocate cacheable slot is drained, misses in the cache, and is not full, the store buffer sends a request to the master ports to complete the cache line. The master port that is ready then sends a read request on AXI and provides data to the store buffer in return. When the slot is full, it can be allocated into the cache. If the respective master port receives a DECERR or SLVERR response during the read transfer, the allocation to the cache is not performed.

2.5.3 Hazards

If a hazard is sent by the L1 masters across read and write channels of slave ports, it can result in unpredictable behavior, as described in the *AMBA AXI Protocol*.

The cache controller must manage a certain number of hazards for which it is responsible. Hazards can occur when data is present in the cache system, that is, one of the buffers, but not yet present in the cache RAM or L3. The hazard checking in the cache controller is performed on bits [20:5] or [31:5] of the address depending on the chosen configuration, see *Cache configurability* on page 2-2.

For example, hazard checking occurs when an address on one of the slave ports matches an address in one of the buffers.

The following hazards can be managed by the cache controller:

- cacheable read when in another read slot of the same slave port
- cacheable read when in another read slot of the other slave port, in configuration with two slave ports
- cacheable read when in Store Buffer
- cacheable read when in Eviction Buffer
- cacheable read when in one read slot of the master ports
- cacheable write when in one read slot of the master ports

- cacheable write when in Eviction Buffer.

Note

The cache controller hazard checking mechanism relies on consistent cacheable attributes for all addresses. In particular, the cache controller does not support checking hazards between transactions targeting the same address but not having the same cacheable attributes.

2.5.4 Cortex-A9 optimizations

The L2CC (PL310) implements several optimizations that the Cortex-A9 processor can use. These optimizations are:

- early AXI write response on slave ports
- prefetch hints
- full line of zero.

Early write response

The AXI protocol specifies that the write response can only be sent back to an AXI master when the last write data has been accepted. This optimization enables the L2CC (PL310) to send the write response of certain write transactions as soon as the store buffer accepts the write address. This behavior is not compatible with the AXI protocol and is disabled by default. You enable this optimization by setting the **Early BRESP Enable** bit in the Auxiliary Control Register (bit [30]). The L2CC (PL310) slave ports then send an early write response only if the input signal AWUSERSx[11] (x=0 or 1) is set to 1'b1 for the corresponding write transaction.

Prefetch hints

When you configure the Cortex-A9 processor to run in SMP mode, see the Cortex-A9 TRM, the automatic data prefetchers (implemented in one or more CPUs) issues special read accesses to the L2CC (PL310). These special reads are called Prefetch Hints. They are indicated when a device sets ARUSERSx[8] to 1. When the L2CC (PL310) slave ports receive such prefetch hints, they do not send any data back to the Cortex-A9 processor, they allocate the targeted cache line into the L2 cache in case of miss. This behavior is not compatible with AXI. When a master other than a Cortex-A9 processor drives the L2CC (PL310) ARUSERSx[8] must be tied 0.

Full line of zero

When the L2CC (PL310) AXI slave ports receive a write transaction with `AWUSERSx[10]`, it indicates that the write actually targets a whole cache line and that all data of this cache line must be reset to zero. The Cortex-A9 processor is likely to use this feature when a CPU is executing a memset routine to initialise a particular memory area. When the L2CC (PL310) receives such a write transaction it ignores the AXI attributes attached to the transaction (size, length, data, and strobes for example) because the whole cache line must be reset. This behavior is not compatible with the AXI protocol, it is disabled by default. You can enable it by setting the Full Line of Zero Enable bit of the Auxiliary Control Register (bit [0]). This behavior also relies on an enable bit in the Cortex-A9 processor. You must take care if you enable this feature because correct behavior relies on consistent enabling in both the Cortex-A9 processor and the L2CC (PL310).

Note

The cache controller does not support Strongly Ordered write accesses with `AWUSERSx[10]` set HIGH.

2.5.5 External error support for L3 memory

The cache controller gives support for sending L3 responses using the response lines of the AXI protocol back to the processor that initiated the transaction. There are several methods to send external error responses created by the L3.

The AXI protocol does not provide a method for passing back an error response that is not combined with its original transaction.

The support provided enables the L1 master core to detect all L3 external aborts, as precise aborts or as imprecise aborts through the interrupt lines.

If L3 detects a security mismatch, it responds with a DECERR response.

External error support is as follows:

Write transactions

If the transfer is a non-bufferable write, an error response is generated by L3, and is returned to the L1 master.

If the transfer is a bufferable write, an OKAY response is given by the cache controller.

Read transactions

The response is attached to the returned data. The AXI read channel returns a response for every transfer of data returned. In the case of an error returned for a Read Allocate line fill, the line is not loaded into the data RAM, the tag RAM is not updated, and the corresponding **SLVERRINTR** or **DECERRINTR** interrupt line is raised.

Evictions or store buffer drain

If the request came from an eviction or store buffer drain to L3, the error response cannot be passed back to L1 because the cache controller no longer has the transaction outstanding in the slave port.

If the write response value is an error then it is returned as an interrupt using the **DECERRINTR** or **SLVERRINTR** signals.

It is recommended that you connect these interrupt sources through an interrupt controller to the processor.

Error response summary

Table 2-20 shows error responses for all combinations of L3 access.

Table 2-20 Error responses for all combinations of L3 access

Access type	Error response mechanism
Noncacheable read transactions and non-bufferable write transactions	Error response is precise and is passed back to the processor using a read response on the slave port read channel or the write response channel.
Linefills associated with a RA	Error response is passed back to processor using a read response on the slave port read channel and an interrupt, the master port cannot distinguish whether the error response is because of the data read, required by slave, or the data required to fill the line. Data is not allocated.
Evictions and store buffer drains	All error responses are imprecise and are passed back to the processor using the interrupt lines.

Note

Evictions, WA linefill, Write-throughs, RA linefills and some RAM and parity errors are the operations that use the **SLVERRINTR** or **DECERRINTR** interrupt lines.

2.5.6 Cache event monitoring

The cache controller supplies pins for event monitoring of the L2 cache. The signals on the pins are held HIGH for one cycle each time the event occurs.

An additional input signal, **SPNIDEN**, configures the level of debug where:

- SP for Secure Privileged
- NI for Non-Invasive, for example trace and performance monitoring
- DEN for Debug Enable.

When the signal on the **SPNIDEN** pin is LOW the event bus and event counters only output or count non-secure events.

When the signal on the **SPNIDEN** pin is HIGH the event bus and event counters output or count non-secure and secure events.

You can poll **SPNIDEN** through the SPNIDEN bit in the Debug Control Register. You must perform a cache sync operation before debug or any analysis that relies on this signal. Synchronizers for the signal are added to prevent any issues from asynchronous domain control.

The event monitoring bus is enabled by writing to the event monitoring bus enable bit in the Auxiliary Control Register. Table 2-20 on page 2-43 shows the event pins.

Table 2-21 Event pins

Pin	Description
CO	Eviction (CastOUT) of a line from the L2 cache.
DRHIT	Data read hit in the L2 cache.
DRREQ	Data read lookup to the L2 cache. Subsequently results in a hit or miss.
DWHIT	Data write hit in the L2 cache.
DWREQ	Data write lookup to the L2 cache. Subsequently results in a hit or miss.
DWTREQ	Data write lookup to the L2 cache with Write-Through attribute. Subsequently results in a hit or miss.
IRHIT	Instruction read hit in the L2 cache.
IRREQ	Instruction read lookup to the L2 cache. Subsequently results in a hit or miss.
SPNIDEN	Secure privileged non-invasive debug enable.
PF	Prefetch linefill sent to L3. This counts both internally generated prefetch accesses and prefetch hints.
WA	Allocation into the L2 cache caused by a write (with Write-Allocate attribute) miss.

2.5.7 Cache interrupts outputs

Table 2-22 shows the interrupt outputs. These outputs provide a sticky output for an interrupt controller to read and generate an interrupt to the processor.

The **L2CCINTR** is a combined interrupt that provides an OR of the nine individual interrupt lines.

Table 2-22 Interrupts

Pin	Description
DECERRINTR	Decode error received on master ports from L3
SLVERRINTR	Slave error received on master ports from L3
ERRRDINTR	Error on L2 data RAM read
ERRRTINTR	Error on L2 tag RAM read
ERRWDINTR	Error on L2 data RAM write
ERRWTINTR	Error on L2 data RAM write
PARRDINTR	Parity error on L2 data RAM read
PARRTINTR	Parity error on L2 tag RAM read
ECNTRINTR	Event Counter Overflow/Increment
L2CCINTR	L2CC Combined Interrupt Output

See *Interrupt Registers* on page 3-20 for more information about these outputs.

2.5.8 Parity and RAM error support

Figure 2-15 on page 2-46 shows cache controller parity and RAM error support. The cache controller generates the parity write data for the data and tag RAMs. For:

Data RAMs The cache controller generates parity write data on a per-byte basis.

Tag RAMs The cache controller generates one parity bit that must be routed to all tag RAMs.

Because only one tag RAM is written at any one time, only one bit is required.

In addition to parity error detection, there are error inputs on the RAM interface, one from Data, **DATAERR** and eight from tag RAM, **TAGERR**. You can use them to identify read and write errors from the RAMs. Those errors are treated in the same way as parity errors.

If a parity error occurs on tag or data RAM during AXI read transactions, a **SLVERR** response is reported back to **RRESPSx** and through the event bus.

If a parity error occurs on tag or data RAM during AXI write transactions, a **SLVERR** response is reported back through an interrupt on the **SLVERRINTR** line.

For cache maintenance operations, if a parity error occurs on tag or data RAM, the error is reported back through the interrupt lines only.

Figure 2-15 shows the cache controller parity and RAM error support.

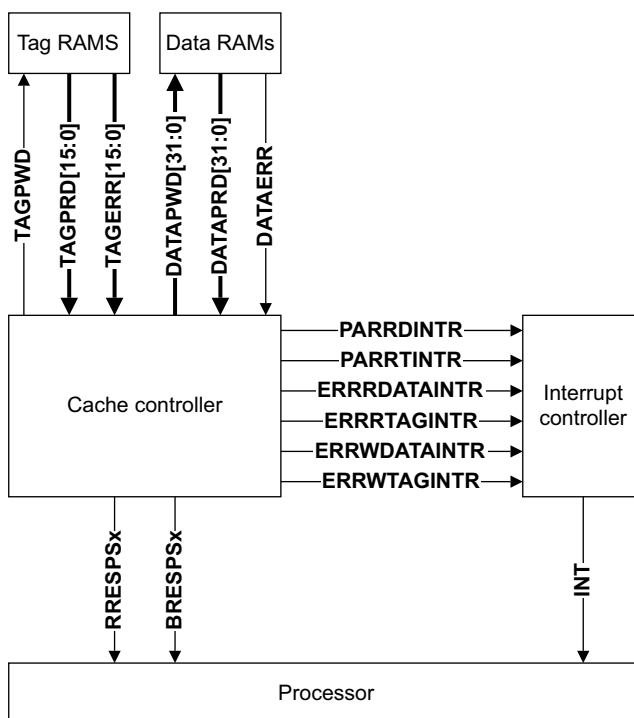


Figure 2-15 Parity and RAM error support for a 16-way implementation

Parity and RAM errors can be reported to the processor through the interrupt lines and/or the **RRESPSx** or **BRESPSx** signals depending on the fault and the transaction that causes that fault. The following sections describe all cases and their effects:

- *Cacheable read requests on AXI slave ports on page 2-47*

- *Write access from the store buffer*
- *AXI master or store buffer allocation requests* on page 2-48
- *Clean maintenance* on page 2-48
- *Invalidate maintenance operation* on page 2-49
- *Clean and Invalidate maintenance operation* on page 2-49.

Cacheable read requests on AXI slave ports

Table 2-23 shows the error signalling cases and effects when there are cacheable read requests on the AXI slave ports.

Table 2-23 Cacheable read requests on AXI slave ports

Case	Effect
Tag parity or RAM error on tag read	An error is flagged through the PARRTINTR/ERRRTINTR interrupt signals.
Data RAM error on data read	An error is flagged through the PARRTINTR/ERRRTINTR interrupt signals.

Write access from the store buffer

Table 2-24 shows the error signalling cases and effects when there are write-through/write-back write accesses from the store buffer.

Table 2-24 Write-through/write-back write access from store buffer

Case	Effect
Tag parity or RAM error on tag read	An error is flagged through the PARRTINTR/ERRRTINTR interrupt signals. No attempt is made to write the cache.
Data RAM error on data write	An error is flagged through the ERRWDINTR interrupt signal. The line is marked as dirty in case of write-back access. Subsequent reads to the same lines are unpredictable.

AXI master or store buffer allocation requests

Table 2-25 shows the error signalling cases and effects when there are AXI master or write-allocate buffer allocation requests.

Table 2-25 AXI M0 and AXI M1 masters or store buffer allocation requests

Case	Effect
Tag parity or RAM error on tag read, for next victim selection	An error is flagged through the PARRTINTR/ERRRTINTR interrupt signals. The allocation is not done.
Data parity or RAM error on data read for eviction	An error is flagged through the PARRDINTR/ERRRDINTR interrupt signals. The allocation process is not stopped and the eviction does not occur in case of data RAM error.
Tag RAM error on tag write	An error is flagged through the ERRWTINTR interrupt signal. Subsequent cache lookups to the same index are unpredictable.
Data RAM error on data write	An error is flagged through the ERRWDINTR interrupt signal. Subsequent reads to that line are unpredictable.

Clean maintenance

Table 2-26 shows the clean maintenance operation error signalling cases and effects.

Table 2-26 Clean maintenance operation cases

Case	Effect
Tag parity or RAM error on tag read	An error is flagged through the PARRTINTR/ERRRTINTR interrupt signal. The clean operation is canceled.
Data parity or RAM error on data read	An error is flagged through the PARRDINTR/ERRRDINTR interrupt signals. The clean operation is done. The eviction does not occur in case of data RAM error.

Invalidate maintenance operation

Table 2-27 shows the Invalidate maintenance operation error signalling cases and effects.

Table 2-27 Invalidate maintenance operation cases

Case	Effect
Tag parity or RAM error on tag read	An error is flagged through the PARRTINTR/ERRTRINTR interrupt signals. The invalidate operation is cancelled.
Tag RAM error on valid information write	An error is flagged through the ERRWTINTR interrupt signal. Subsequent reads to that line are unpredictable.

Clean and Invalidate maintenance operation

Table 2-28 shows the Clean and Invalidate maintenance operation error signalling cases and effects.

Table 2-28 Clean and Invalidate maintenance operation cases

Case	Effect
Tag parity or RAM error on tag read	An error is flagged through the PARRTINTR/ERRRTINTR interrupt signals. The operation is canceled.
Data parity or RAM error on data read	An error is flagged through the PARRTINR/ERRRDINTR interrupt signals. The operation is done. The eviction does not occur in case of data RAM error.
Tag RAM error on valid information write	An error is flagged through the ERRWTINTR interrupt signal. Subsequent reads to that line are unpredictable.

2.6 Power modes

Power modes are controlled by clock, reset, and power management blocks within the system. You have to write additional software to save the settings of registers, so that they can be restored to the same state at a later time. Power modes can be:

- *Run mode*
- *Standby mode*
- *Dormant mode* on page 2-51
- *Shutdown mode* on page 2-51.

2.6.1 Run mode

This is the normal mode of operation in which all cache controller functionality is available.

2.6.2 Standby mode

Standby mode consists of externally disabling the clock of the device, while keeping the design powered up. This reduces the power drawn to the static leakage current.

If you want to put a whole system, including the cache controller, into standby mode, you must first perform an L2 Cache Sync operation to ensure the memory system is not affected by the low power mode entry. You must also ensure that you only put the cache controller into standby mode when the L1 masters have been put into standby wait for interrupt mode. The cache controller can be put into standby mode when its **IDLE** output signal is asserted.

On exit from standby mode, the cache controller clock must be restarted as soon as the L1 masters wake up and send their first accesses to the cache controller. Ensure that the cache controller returns to run mode before returning active clock edges to the L1 masters.

To reduce power consumption, you might want to externally stop the clock driving the cache controller when there is no activity in the cache controller. This idle state is indicated when the output signal **IDLE** is asserted. When stopped, the clock must be restarted as soon as a transaction is initiated by the L1 masters on the AXI slave port(s) of the cache controller, that is, in the same cycle as **ARVALIDSx**, **AWVALIDSx**, or **WVALIDSx** are asserted. When re-started, the clock driving the cache controller must remain active for several clock cycles even if the **IDLE** signal remains asserted. This allows the system to capture the new transaction and also allows the non-idle state to propagate to the **IDLE** output.

2.6.3 Dormant mode

Dormant mode enables the cache controller to be powered down, leaving the cache memories powered up and maintaining their state. This mode eliminates any power being drawn by the standard cells used to implement the cache controller.

Because the store buffer and the internal register banks are implemented in standard cells, you must save the values of the programmed registers before power can be removed from the cache controller. You must therefore perform the following software operations before entering dormant mode:

1. Save all cache controller configuration registers.
2. Perform a cache sync operation.
3. Wait for **IDLE** signal to be asserted.
4. Stop clock and remove power.
5. Restore clock and reset the cache controller.
6. Restore configuration registers.

As with standby mode, the cache controller can go into dormant mode only when the L1 masters are inactive and issuing no more transactions. Transitioning from Dormant mode to Run mode is triggered by the external power controller, which must assert reset. Ensure the cache controller is placed back into run mode prior to returning active clock edges to the L1 masters.

In Dormant mode, the RAMs must be kept powered up while the cache controller is powered down. Because of this, clamping cells must be implemented between the RAMs and the cache controller.

———— **Note** ————

These cells are not part of the cache controller.

When the RAMs exit from their retention state, they can indicate to the cache controller logic that they are not ready to accept accesses, by asserting their wait signal.

2.6.4 Shutdown mode

Shutdown mode powers down the entire device, including the cache controller logic and the L2 cache RAMs. You must save all states externally, including cleaning any dirty data that might exist in the cache memory. You can return the cache controller to run mode by asserting reset.

Chapter 3

Programmers Model

This chapter describes the programmers model. It contains the following sections:

- *About this programmers model* on page 3-2
- *Register summary* on page 3-4
- *Register descriptions* on page 3-8.

3.1 About this programmers model

The following applies to the registers used in the cache controller:

- The base address of the cache controller is not fixed, and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.

The cache controller is controlled through a set of memory-mapped registers that occupy a re-locatable 4KB of memory. You must define this region with Strongly Ordered or Device memory attributes in the L1 page tables. You can access the registers through direct address decoding in the slave ports.

REGFILEBASE[31:12] provides the base address for these ports.

- Reserved or unused address locations must not be accessed because this can result in unpredictable behavior of the device.
- All registers support read and write accesses unless otherwise stated in the relevant text. A write updates the contents of a register and a read returns the contents of the register.
- All writes to registers automatically perform an initial Cache Sync operation before proceeding.

When accessing the registers:

- Bits [1:0] of the address must be zero, otherwise a SLVERR response is returned.
- The burst length must be equal to zero, otherwise a SLVERR response is returned.
- Only 32-bit accesses are allowed, otherwise a SLVERR response is returned.
- Exclusive accesses are not allowed. A SLVERR response is returned.
- The cache controller ignores the write strobes and always considers the write strobes to be 0x0f or 0xf0 depending on the offset. When the cache controller accesses the registers it does not support sparse write strobes.
- Any write to a writable register returns SLVERR while a background operation is in progress.

3.1.1 Initialization sequence

Caution

At boot time you must perform a Secure write to the Invalidate by Way, offset 0x77C, to invalidate all entries in the cache.

As an example, a typical cache controller start-up programming sequence consists of the following register operations:

1. Write to the Auxiliary, Tag RAM Latency, and Data RAM Latency Control registers using a read-modify-write to set up global configurations:
 - Associativity, Way Size
 - Latencies for RAM accesses.
2. Secure write to the Invalidate by Way, offset 0x77C, to invalidate all entries in cache:
 - Write 0xFFFF to 0x77C
 - Poll cache maintenance register until invalidate operation is complete.
3. Write to the Lockdown D and Lockdown I Register 9 if required.
4. Write to interrupt clear register to clear any residual raw interrupts set.
5. Write to the Interrupt Mask Register if you want to enable interrupts.
6. Write to Control Register 1 with the LSB set to 1 to enable the cache.

If you write to the Auxiliary, Tag RAM Latency, or Data RAM Latency Control Register with the L2 cache enabled, this results in a SLVERR. You must disable the L2 cache by writing to the Control Register 1 before writing to the Auxiliary, Tag RAM Latency, or Data RAM Latency Control Register.

3.2 Register summary

Table 3-1 shows the register map for the cache controller.

Table 3-1 Cache controller register map

Offset range	Reads	Writes	Secure
0x000 - 0x0FC	Cache ID and Cache Type	Ignored	NS/S
0x100 - 0x1FC	Control	Control	Write S Read NS/S
0x200 - 0x2FC	Interrupt/Counter Control Registers	Interrupt/Counter Control Registers	NS/S
0x300 - 0x6FC	Reserved	Reserved	-
0x700 - 0x7FC	Cache Maintenance Operations	Cache Maintenance Operations	Secure bit of access affects operation
0x800 - 0x8FC	Reserved	Reserved	-
0x900 - 0x9FC	Cache Lockdown	Cache Lockdown	Secure bit of access affects operation
0xA00 - 0xBFC	Reserved	Reserved	-
0xC00 - 0xCFC	Address Filtering	Address Filtering	Write S Read NS/S
0xD00 - 0xEFC	Reserved	Reserved	-
0xF00 - 0xFFC	Debug and Prefetch	Debug and Prefetch	Write S Read NS/S

All register addresses in the cache controller are fixed relative to the base address. Table 3-2 on page 3-5 shows the registers in base offset order.

Table 3-2 Summary of cache controller registers

Offset	Name	Type	Reset	Width	Description
0x000	reg0_cache_id	RO	0x410000C4 ^a	32	Cache ID Register on page 3-8
0x004	reg0_cache_type	RO	0x1C100100 ^b	32	Cache Type Register on page 3-9
0x100	reg1_control	RW	0x00000000	32	Control Register on page 3-10
0x104	reg1_aux_control	RW	0x02020000 ^b	32	Auxiliary Control Register on page 3-12
0x108	reg1_tag_ram_control	RW	0x00000nnn ^c	32	Tag and Data RAM Latency Control Registers on page 3-15
0x10C	reg1_data_ram_control	RW	0x00000nnn ^d	32	
0x200	reg2_ev_counter_ctrl	RW	0x00000000	32	Event Counter Control Register on page 3-17
0x204	reg2_ev_counter1_cfg	RW	0x00000000	32	Event Counter Configuration Registers on page 3-18
0x208	reg2_ev_counter0_cfg	RW	0x00000000	32	
0x20C	reg2_ev_counter1	RW	0x00000000	32	Event Counter Value Registers on page 3-20
0x210	reg2_ev_counter0	RW	0x00000000	32	
0x214	reg2_int_mask ^e	RW	0x00000000	32	Interrupt Registers on page 3-20
0x218	reg2_int_mask_status ^e	RO	0x00000000	32	
0x21C	reg2_int_raw_status ^e	RO	0x00000000	32	
0x220	reg2_int_clear ^e	WO	0x00000000	32	
0x730	reg7_cache_sync	RW	0x00000000	32	Cache Maintenance Operations on page 3-26
0x770	reg7_inv_pa	RW	0x00000000	32	
0x77C	reg7_inv_way	RW	0x00000000	32	
0x7B0	reg7_clean_pa	RW	0x00000000	32	
0x7B8	reg7_clean_index	RW	0x00000000	32	
0x7BC	reg7_clean_way	RW	0x00000000	32	
0x7F0	reg7_clean_inv_pa	RW	0x00000000	32	
0x7F8	reg7_clean_inv_index	RW	0x00000000	32	
0x7FC	reg7_clean_inv_way	RW	0x00000000	32	

Table 3-2 Summary of cache controller registers (continued)

Offset	Name	Type	Reset	Width	Description
0x900	reg9_d_lockdown0	RW	0x00000000	32	<i>Cache Lockdown</i> on page 3-33
0x904	reg9_i_lockdown0	RW	0x00000000	32	
0x908	reg9_d_lockdown1 ^f	RW	0x00000000	32	
0x90C	reg9_i_lockdown1 ^f	RW	0x00000000	32	
0x910	reg9_d_lockdown2 ^f	RW	0x00000000	32	
0x914	reg9_i_lockdown2 ^f	RW	0x00000000	32	
0x918	reg9_d_lockdown3 ^f	RW	0x00000000	32	
0x91C	reg9_i_lockdown3 ^f	RW	0x00000000	32	
0x920	reg9_d_lockdown4 ^f	RW	0x00000000	32	
0x924	reg9_i_lockdown4 ^f	RW	0x00000000	32	
0x928	reg9_d_lockdown5 ^f	RW	0x00000000	32	
0x92C	reg9_i_lockdown5 ^f	RW	0x00000000	32	
0x930	reg9_d_lockdown6 ^f	RW	0x00000000	32	
0x934	reg9_i_lockdown6 ^f	RW	0x00000000	32	
0x938	reg9_d_lockdown7 ^f	RW	0x00000000	32	
0x93C	reg9_i_lockdown7 ^f	RW	0x00000000	32	
0x950	reg9_lock_line_en ^g	RW	0x00000000	32	<i>Address Filtering</i> on page 3-39
0x954	reg9_unlock_way ^g	RW	0x00000000	32	
0xC00	reg12_addr_filtering_start	RW	0x00000000 ^h	32	
0xC04	reg12_addr_filtering_end	RW	0x00000000 ⁱ	32	<i>Debug Register</i> on page 3-40
0xF40	reg15_debug_ctrl	RW	0x00000000	32	
0xF60	Prefetch Offset Register	RW	0x00000000	32	

- a. This value is pin dependent, depending on how external **CACHEID** pins are tied.
- b. This value is pin dependent, depending on how external **WAYSIZ** and **ASSOCIATIVITY** pins are tied.
- c. This value depends on the chosen values for **pl310_TAG_SETUP_LAT**, **pl310_TAG_READ_LAT** and **pl310_TAG_WRITE_LAT** parameters.

- d. This value depends on the chosen values for `pl310_DATA_SETUP_LAT`, `pl310_DATA_READ_LAT` and `pl310_DATA_WRITE_LAT` parameters.
- e. The cache interrupt registers are those that can be accessed by secure and non-secure operations.
- f. These registers are implemented if the option `pl310_LOCKDOWN_BY_MASTER` is enabled. Otherwise, they are unused.
- g. These registers are implemented if the option `pl310_LOCKDOWN_BY_LINE` is enabled. Otherwise, they are unused.
- h. This value is pin dependent if address filtering is implemented, depending on how external **CFGADDRFILTEN** and **CFGADDRFILSTART** pins are tied.
- i. This value is pin dependent if address filtering is implemented, depending on how external **CFGADDRFILTEEND** pins are tied.

3.3 Register descriptions

This section describes the cache controller registers. Table 3-2 on page 3-5 provides cross references to individual registers.

3.3.1 Cache ID Register

The reg0_cache_id Register characteristics are:

- Purpose**
Returns the 32-bit device ID code it reads off the **CACHEID** input bus. The value is specified by the system integrator.
- Usage constraints**
There are no usage constraints.
- Configurations**
Available in all configurations.
- Attributes**
See the register summary in Table 3-2 on page 3-5.

Figure 3-1 shows the reg0_cache_id Register bit assignments.

31	24	23	16	15	10	9	6	5	0
Implementer				Reserved		CACHE ID		Part number	RTL release

Figure 3-1 reg0_cache_id Register bit assignments

Table 3-3 shows the reg0_cache_id register bit assignments.

Table 3-3 reg0_cache_id Register bit assignments

Bits	Field	Description
[31:24]	Implementer	0x41 (ARM)
[23:16]	Reserved	SBZ
[15:10]	CACHE ID	-
[9:6]	Part number	0x3
[5:0]	RTL release	0x4

- Note**
- Part number 0x3 denotes PrimeCell Level 2 Cache Controller (PL310)
 - RTL release 0x4 denotes r2p0 code of the cache controller. See the Release Note for the value of these bits for other releases.

3.3.2 Cache Type Register

The reg0_cache_type Register characteristics are:

Purpose Returns the 32-bit cache type.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-5.

Figure 3-2 shows the reg0_cache_type register bit assignments.

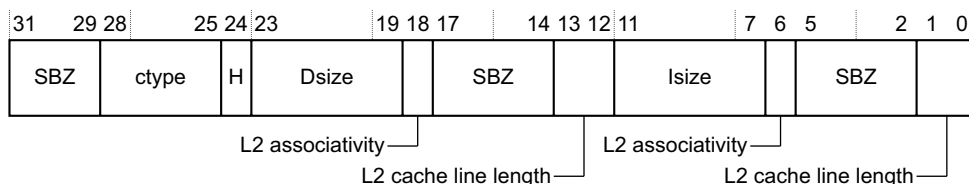


Figure 3-2 reg0_cache_type Register bit assignments

Table 3-4 shows the reg0_cache_type Register bit assignments.

Table 3-4 reg0_cache_type Register bit assignments

Bits	Field	Sub-field	Comments
[31:29]	SBZ		000
[28:25]	ctype		11xy, where: x=1 if p1310_LOCKDOWN_BY_MASTER is defined, otherwise 0 y=1 if p1310_LOCKDOWN_BY_LINE is defined, otherwise 0. See <i>Cache Lockdown</i> on page 3-33 for information on the lockdown features.
[24]	H		0 - unified ^a
[23:19]	Dsize		-
[23]		SBZ/RAZ	0
[22:20]		L2 cache way size	Read from Auxiliary Control Register [19:17]
[19]		SBZ/RAZ	0
[18]	L2 associativity		Read from Auxiliary Control Register [16]

Table 3-4 reg0_cache_type Register bit assignments (continued)

Bits	Field	Sub-field	Comments
[17:14]	SBZ		0
[13:12]	L2 cache line length		00 - 32 bytes
[11:7]	Isize		-
	[11]	SBZ/RAZ	0
	[10:8]	L2 cache way size	Read from Auxiliary Control Register [19:17]
	[7]	SBZ/RAZ	0
[6]	L2 associativity		Read from Auxiliary Control Register [16]
[5:2]	SBZ		0
[1:0]	L2 cache line length		00 - 32 bytes

a. 1 = Harvard.

The Cache Type Register returns the 32-bit cache type. This register provides data for cache type, cache size, way size, associativity and cache line length, in instruction and data format. The cache size is a product of:

- L2 cache way size
- L2 associativity.

3.3.3 Control Register

The reg1_control Register characteristics are:

Purpose	Enables or disables the cache controller.
Usage constraints	This register enables or disables the cache controller. Must be written using a secure access. It can be read using either a secure or a NS access. Writing to this register with a NS access causes a write response signal with a DECERR response, and the register is not updated, only permitting a secure access to enable or disable the cache controller.

When receiving a transaction to enable or disable the cache by modifying this register the cache controller follows the described sequence. This prevents any unpredictable behavior if there are subsequent writes to any of the L2 registers.

1. Lock slave ports and wait for all outstanding transactions to complete and all buffers to be empty by performing a cache sync.
2. Update register.
3. Return write response.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-5.

Figure 3-3 shows the reg1_control Register bit assignments.

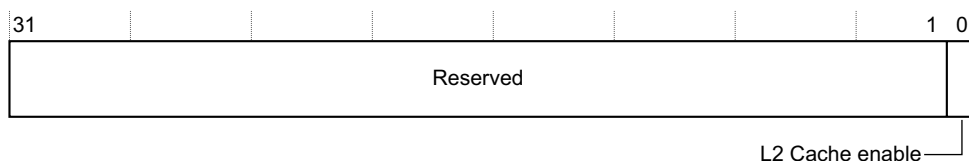


Figure 3-3 reg1_control Register bit assignments

Table 3-5 shows the reg1_control register bit assignments.

Table 3-5 reg1_control Register bit assignments

Bits	Field	Description
[31:1]	Reserved	SBZ/RAZ
[0]	L2 Cache enable	0 = L2 Cache is disabled. This is the default value. 1 = L2 Cache is enabled.

3.3.4 Auxiliary Control Register

The reg1_aux_control Register characteristics are:

- Purpose

Configures:

 - cache behavior
 - event monitoring
 - way size
 - associativity.

Usage constraints The register must be written to using a secure access, and it can be read using either a secure or a NS access. If you write to this register with a NS access, it results in a write response with a DECERR response, and the register is not updated. Writing to this register with the L2 cache enabled, that is, bit[0] of L2 Control Register set to 1, results in a SLVERR. The DECERR response has priority over SLVERR.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-5.

Figure 3-4 shows the reg1_aux_control Register bit assignments.

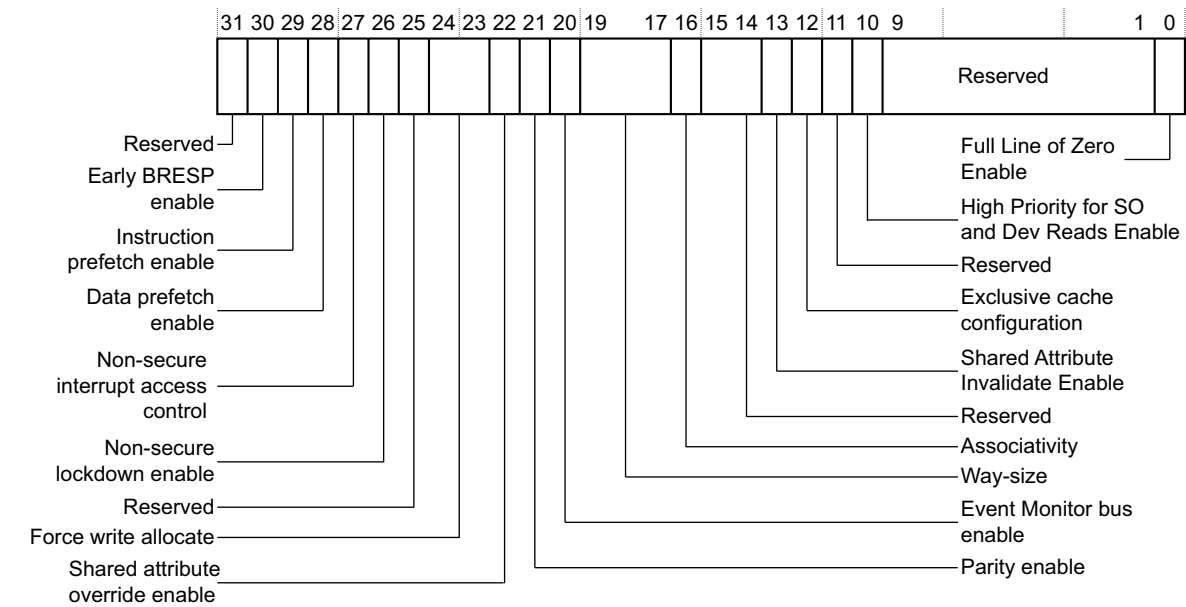


Figure 3-4 reg1_aux_control Register bit assignments

Table 3-6 shows the reg1_aux_control Register bit assignments.

Table 3-6 reg1_aux_control Register bit assignments

Bits	Field	Description
[31]	Reserved	SBZ/RAZ
[30]	Early BRESP enable	0 = Early BRESP disabled. This is the default. 1 = Early BRESP enabled. See <i>Early write response</i> on page 2-41.
[29]	Instruction prefetch enable ^a	0 = Instruction prefetching disabled. This is the default. 1 = Instruction prefetching enabled.
[28]	Data prefetch enable ^a	0 = Data prefetching disabled. This is the default. 1 = Data prefetching enabled.
[27]	Non-secure interrupt access control	0 = Interrupt Clear (0x220) and Interrupt Mask (0x214) can only be modified or read with secure accesses. This is the default. 1 = Interrupt Clear (0x220) and Interrupt Mask (0x214) can be modified or read with secure or non-secure accesses.
[26]	Non-secure lockdown enable	0 = Lockdown registers cannot be modified using non-secure accesses. This is the default. 1 = Non-secure accesses can write to the lockdown registers.
[25]	Reserved	SBO/RAO
[24:23]	Force write allocate	00 = Use AWCACHE attributes for WA. This is the default. 01 = Force no allocate, set WA bit always 0. 10 = Override AWCACHE attributes, set WA bit always 1 (all cacheable write misses become write allocated). 11 = Internally mapped to 00. See <i>Cache operation</i> on page 2-13 for more details.
[22]	Shared attribute override enable	0 = Treats shared accesses as specified in <i>Shared attribute</i> on page 2-19. This is the default. 1 = Shared attribute internally ignored.
[21]	Parity enable	0 = Disabled. This is the default. 1 = Enabled.
[20]	Event monitor bus enable	0 = Disabled. This is the default. 1 = Enabled.

Table 3-6 reg1_aux_control Register bit assignments (continued)

Bits	Field	Description
[19:17]	Way-size ^b	000 = Reserved, internally mapped to 16KB. 001 = 16KB. 010 = 32KB. 011 = 64KB. 100 = 128KB. 101 = 256KB. 110 = 512KB. 111 = Reserved, internally mapped to 512 KB.
[16]	Associativity ^c	0 = 8-way. 1 = 16-way.
[15:14]	Reserved	SBZ/RAZ
[13]	Shared Attribute Invalidate Enable	0 = Shared invalidate behavior disabled. This is the default. 1 = Shared invalidate behavior enabled (if Shared Attribute Override Enable bit not set). See <i>Shared attribute</i> on page 2-19.
[12]	Exclusive cache configuration	0 = Disabled. This is the default. 1 = Enabled, see <i>Exclusive cache configuration</i> on page 2-21.
[11]	Reserved	SBZ/RAZ
[10]	High Priority for SO and Dev Reads Enable	0 = Strongly Ordered and Device reads have lower priority than cacheable accesses when arbitrated in the L2CC (PL310) master port(s). This is the default. 1 = Strongly Ordered and Device reads get the highest priority when arbitrated in the L2CC (PL310) master port(s).
[9:1]	Reserved	SBZ/RAZ
[0]	Full Line of Zero Enable	0 = Full Line of Zero behavior disabled. This is the default. 1 = Full Line of Zero behavior Enabled.

- When enabled, as soon as a cacheable read transaction is received by one of the slave ports, a cache lookup is done on the following cache line. In case of miss, the cache line is prefetched from L3 and allocated into the L2 cache. The prefetch mechanism is not launched in the case of 4KB boundary crossing.
- The default value of the way size depends on how the external WAYSIZE pins are tied.
- The default value of the associativity depends on how the external ASSOCIATIVITY pin is tied, and how the RTL is configured.

3.3.5 Tag and Data RAM Latency Control Registers

The reg1_tag_ram_control and reg1_data_ram_control Register characteristics are:

Purpose

Configures:

- Tag RAM latencies for the Tag RAM Latency Control Register
- Data RAM latencies for the Data RAM Latency Control Register.

Usage constraints

The registers must be written using a secure access. They can be read using either a secure or a NS access. If you write to these registers with a NS access, it results in a write response with a DECERR response, and the registers are not updated. Writing to these registers with the L2 cache enabled, that is, bit[0] of the Control Register set to 1, results in a SLVERR.

Configurations

Available in all configurations.

Attributes

See the register summary in Table 3-2 on page 3-5.

Figure 3-5 shows the reg1_tag_ram_control and reg1_data_ram_control Register bit assignments.

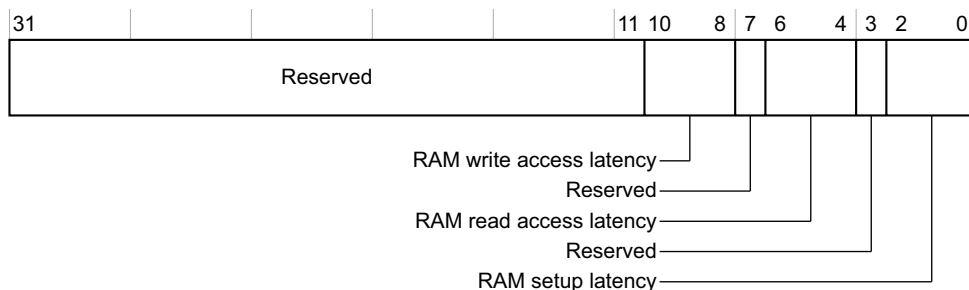


Figure 3-5 reg1_tag_ram_control and reg1_data_ram_control Register bit assignments

Table 3-7 shows the reg1_tag_ram_control and reg1_data_ram_control Register bit assignments.

Table 3-7 reg1_tag_ram_control and reg1_data_ram_control Register bit assignments

Bits	Field	Description
[31:11]	Reserved	SBZ/RAZ
[10:8]	RAM write access latency	Default value depends on the value of pl310_TAG_WRITE_LAT for reg1_tag_ram_control or pl310_DATA_WRITE_LAT for reg1_data_ram_control. 000 = 1 cycle of latency, there is no additional latency. 001 = 2 cycles of latency. 010 = 3 cycles of latency. 011 = 4 cycles of latency. 100 = 5 cycles of latency. 101 = 6 cycles of latency. 110 = 7 cycles of latency. 111 = 8 cycles of latency.
[7]	Reserved	SBZ/RAZ

Table 3-7 reg1_tag_ram_control and reg1_data_ram_control Register bit assignments (continued)

Bits	Field	Description
[6:4]	RAM read access latency	Default value depends on the value of pl310_TAG_READ_LAT for reg1_tag_ram_control or pl310_DATA_READ_LAT for reg1_data_ram_control. 000 = 1 cycle of latency, there is no additional latency. 001 = 2 cycles of latency. 010 = 3 cycles of latency. 011 = 4 cycles of latency. 100 = 5 cycles of latency. 101 = 6 cycles of latency. 110 = 7 cycles of latency. 111 = 8 cycles of latency.
[3]	Reserved	SBZ/RAZ
[2:0]	RAM setup latency	Default value depends on the value of pl310_TAG_SETUP_LAT for reg1_tag_ram_control or pl310_DATA_SETUP_LAT for reg1_data_ram_control. 000 = 1 cycle of latency, there is no additional latency. 001 = 2 cycles of latency. 010 = 3 cycles of latency. 011 = 4 cycles of latency. 100 = 5 cycles of latency. 101 = 6 cycles of latency. 110 = 7 cycles of latency. 111 = 8 cycles of latency.

3.3.6 Event Counter Control Register

The reg2_ev_counter_ctrl Register characteristics are:

Purpose Permits the event counters to be enabled and reset.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-5.

Figure 3-6 on page 3-18 shows the reg2_ev_counter_ctrl Register bit assignments.

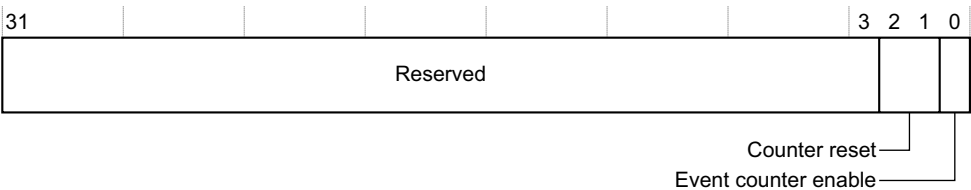


Figure 3-6 reg2_ev_counter_ctrl Register bit assignments

Table 3-8 shows the reg2_ev_counter_ctrl Register bit assignments.

Table 3-8 reg2_ev_counter_ctrl Register bit assignments

Bits	Field	Description
[31:3]	Reserved	SBZ/RAZ
[2:1]	Counter reset	Always Read as zero. Corresponding counters are reset when the bit is written to by 1: Bit 2 = Event Counter1 reset Bit 1 = Event Counter0 reset.
[0]	Event counter enable	0 = Event Counting Disable. This is the default. 1 = Event Counting Enable.

3.3.7 Event Counter Configuration Registers

The reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register characteristics are:

- Purpose** Enables event counter 1 and 0 to be driven by a specific event. Counter 1 or counter 0 increments when the event occurs. *Cache event monitoring* on page 2-44 describes the counter event source signals.
- Usage constraints** There are no usage constraints.
- Configurations** Available in all configurations.
- Attributes** See the register summary in Table 3-2 on page 3-5.

Figure 3-7 on page 3-19 shows the reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments.

Note

When the **SPNIDEN** input pin is LOW the event counters only increment on non-secure events, secure events are not counted unless the **SPNIDEN** pin signal is configured HIGH.

3.3.8 Event Counter Value Registers

The reg2_ev_counter0 and reg2_ev_counter1 Register characteristics are:

Purpose	Enable the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration Registers. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.
Usage constraints	Can only be written to when bits [5:2] of the Event Counter Configuration Registers are set to Counter Disabled.
Configurations	Available in all configurations.
Attributes	See the register summary in Table 3-2 on page 3-5.

Table 3-10 shows the reg2_ev_counter0 and reg2_ev_counter1 Register bit assignments.

Table 3-10 reg2_ev_counter0 and reg2_ev_counter1 Register bit assignments

Bits	Field	Description
[31:0]	Counter value	Total of the event selected.

3.3.9 Interrupt Registers

There are four interrupt registers:

- *Interrupt Mask Register* on page 3-22
- *Masked Interrupt Status Register* on page 3-23
- *Raw Interrupt Status Register* on page 3-24
- *Interrupt Clear Register* on page 3-25.

Figure 3-8 on page 3-21 shows the register bit assignments.

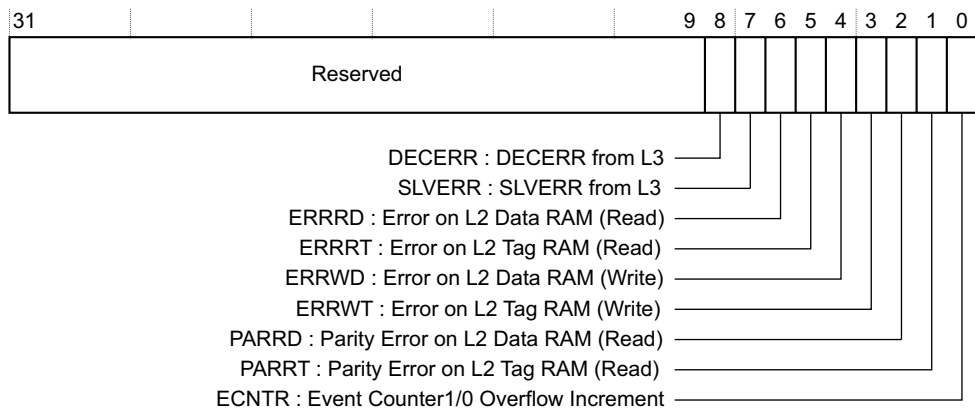


Figure 3-8 Interrupt Register bit assignments

Interrupt Mask Register

The reg2_int_mask Register characteristics are:

- Purpose

This register enables or masks interrupts from being triggered on the external pins of the cache controller. Figure 3-8 on page 3-21 shows the register bit assignments. The bit assignments enables the masking of the interrupts on both their individual outputs and the combined **L2CCINTR** line. Clearing a bit by writing a 0, disables the interrupt triggering on that pin. All bits are cleared by a reset. You must write to the register bits with a 1 to enable the generation of interrupts.
- Usage constraints

Non-secure writes to this register are dependent on Auxiliary Control Register bit [27]. If bit [27] of the Auxiliary Control Register is 1'b0, a Non secure write to the Interrupt Mask Register results in a DECERR response.
- Configurations

Available in all configurations.
- Attributes

See the register summary in Table 3-2 on page 3-5.

Table 3-11 shows the reg2_int_mask Register bit assignments.

Table 3-11 reg2_int_mask Register bit assignments

Bits	Field	Description
[31:9]	Reserved	SBZ/RAZ
[8]	DECERR: DECERR from L3	1 = Enabled.
[7]	SLVERR: SLVERR from L3	0 = Masked. This is the default.
[6]	ERRRD: Error on L2 data RAM (Read)	
[5]	ERRRT: Error on L2 tag RAM (Read)	
[4]	ERRWD: Error on L2 data RAM (Write)	
[3]	ERRWT: Error on L2 tag RAM (Write)	
[2]	PARRD: Parity Error on L2 data RAM (Read)	
[1]	PARRT: Parity Error on L2 tag RAM (Read)	
[0]	ECNTR: Event Counter1/0 Overflow Increment	

Masked Interrupt Status Register

The reg2_int_mask_status Register characteristics are:

Purpose This register is a read-only. It returns the masked interrupt status. This register can be accessed by secure and non-secure operations. The register gives an AND function of the raw interrupt status with the values of the interrupt mask register. All the bits are cleared by a reset. A write to this register is ignored.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-5.

Table 3-12 shows the reg2_int_mask_status Register bit assignments.

Table 3-12 Masked Interrupt Status Register bit assignments

Bits	Field	Description
[31:9]	Reserved	RAZ
[8]	DECERR: DECERR from L3	Bits read can be HIGH or LOW: If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.
[7]	SLVERR: SLVERR from L3	
[6]	ERRRD: Error on L2 data RAM (Read)	If bits read LOW, either no interrupt has been generated or the interrupt is masked.
[5]	ERRRT: Error on L2 tag RAM (Read)	
[4]	ERRWD: Error on L2 data RAM (Write)	
[3]	ERRWT: Error on L2 tag RAM (Write)	
[2]	PARRD: Parity Error on L2 data RAM (Read)	
[1]	PARRT: Parity Error on L2 tag RAM (Read)	
[0]	ECNTR: Event Counter1/0 Overflow Increment	

Raw Interrupt Status Register

The reg2_int_raw_status Register characteristics are:

- Purpose

The Raw Interrupt Status Register enables the interrupt status that excludes the masking logic.
- Usage constraints

There are no usage constraints.
- Configurations

Available in all configurations.
- Attributes

See the register summary in Table 3-2 on page 3-5.

Table 3-13 shows the reg2_int_raw_status Register bit assignments.

Table 3-13 reg2_int_raw_status Register bit assignments

Bits	Field	Description
[31:9]	Reserved	RAZ
[8]	DECERR: DECERR from L3	If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.
[7]	SLVERR: SLVERR from L3	
[6]	ERRRD: Error on L2 data RAM (Read)	If bits read LOW, no interrupt has been generated.
[5]	ERRRT: Error on L2 tag RAM (Read)	
[4]	ERRWD: Error on L2 data RAM (Write)	
[3]	ERRWT: Error on L2 tag RAM (Write)	
[2]	PARRD: Parity Error on L2 data RAM (Read)	
[1]	PARRT: Parity Error on L2 tag RAM (Read)	
[0]	ECNTR: Event Counter1/0 Overflow Increment	

Interrupt Clear Register

The reg2_int_clear Register characteristics are:

- Purpose** Clears the Raw Interrupt Status Register bits.
- Usage constraints** Non-secure access to this register is dependent on Auxiliary Control Register bit [27]. If bit [27] of the Auxiliary Control Register is set to 1'b0, a Non secure write to this register results in a DECERR response. A read to this register returns zero.
- Configurations** Available in all configurations.
- Attributes** See the register summary in Table 3-2 on page 3-5.

Table 3-14 shows the reg2_int_clear Register bit assignments.

Table 3-14 reg2_int_clear Register bit assignments

Bits	Field	Description
[31:9]	Reserved	RAZ
[8]	DECERR: DECERR from L3	When a bit is written as 1, it clears the corresponding bit in the Raw Interrupt Status Register
[7]	SLVERR: SLVERR from L3	
[6]	ERRRD: Error on L2 data RAM (Read)	When a bit is written as 0, it has no effect
[5]	ERRRT: Error on L2 tag RAM (Read)	
[4]	ERRWD: Error on L2 data RAM (Write)	
[3]	ERRWT: Error on L2 tag RAM (Write)	
[2]	PARRD: Parity Error on L2 data RAM (Read)	
[1]	PARRT: Parity Error on L2 tag RAM (Read)	
[0]	ECNTR: Event Counter1/0 Overflow Increment	

3.3.10 Cache Maintenance Operations

The Cache Maintenance Operations registers have different behavior, depending on the AXI security flag of the access requesting a cache operation. If the operation is specific to the Way or Index/Way, their behavior is presented in the following manner:

Secure access

The secure bit of the tag is ignored and the maintenance operation can affect both secure and non-secure lines.

Non-secure access

The secure bit of the tag is checked, a lookup must be done for each non-secure maintenance operation, and the maintenance operation can only affect non-secure lines. Secure lines in cache are ignored and unmodified.

Also, depending on the AXI security flag of the access requesting a cache operation, if the operation is specific to the *Physical Address* (PA), the behavior is presented in the following manner:

Secure access

The data in the cache is only affected by the operation if it is secure.

Non-secure access

The data in the cache is only affected by the operation if it is non-secure.

Table 3-15 shows the cache maintenance operations. They are executed by writing to the Cache Operations Registers. See also Table 3-16 on page 3-29.

Table 3-15 Maintenance operations

Operation	Base offset	Type	Bit assignment format
Cache Sync	0x730	RW	-
Invalidate Line by PA	0x770	RW	See Figure 3-9 on page 3-27
Invalidate by Way	0x77C	RW	See Figure 3-11 on page 3-28
Clean Line by PA	0x7B0	RW	See Figure 3-9 on page 3-27
Clean Line by Index/Way	0x7B8	RW	See Figure 3-10 on page 3-27
Clean by Way	0x7BC	RW	See Figure 3-11 on page 3-28

Table 3-15 Maintenance operations (continued)

Operation	Base offset	Type	Bit assignment format
Clean and Invalidate Line by PA	0x7F0	RW	See Figure 3-9
Clean and Invalidate Line by Index/Way	0x7F8	RW	See Figure 3-10
Clean and Invalidate by Way	0x7FC	RW	See Figure 3-11 on page 3-28

Figure 3-9 shows the PA format.



Figure 3-9 Physical address format

Figure 3-10 shows the Index or Way format.



Figure 3-10 Index/way format

———— **Note** ————

For a 16-way implementation, all four bits [31:28] are used. If the 16-way option is not enabled, bit [31] is reserved.

Atomic operations

The following are atomic operations:

- Clean Line by PA or by Index/Way.
- Invalidate Line by PA.
- Clean and Invalidate Line by PA or by Index/Way.
- Cache Sync.

These operations stall the slave ports until they are complete. When these registers are read, bit [0], the C flag, indicates that a background operation is in progress. When written, bit 0 must be zero.

Background operations

The following operations are run as background tasks:

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

————— **Note** —————

If lockdown by line is implemented, the Unlock All Lines operation is also a background operation.

Writing to the register starts the operation on the Ways set to 1 in bits [15:0]. When a Way bit is set to 1, it is reset to 0 when the corresponding way is totally cleaned or invalidated. You must poll the register to see when the operation is complete, indicated by all bits cleared.

Figure 3-11 shows the Way Format. You can select multiple ways at the same time, by setting the Way bits to 1.

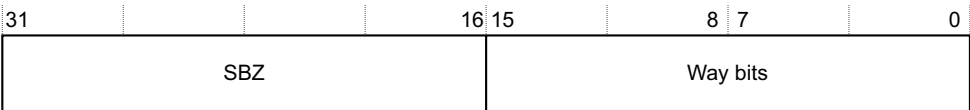


Figure 3-11 Way format

————— **Note** —————

For a 16-way implementation, all bits [15:0] are used. If the 16-way option is not enabled, bits [15:8] are reserved.

During background operations any write to a configuration and control register while a background operation is running results in a SLVERR response.

During background operations, the targeted ways are considered as locked until they have been treated. This means that no allocation occurs to that way on read or write misses. Read or write hits are permitted to access the way. No hazard detection is performed on data returned. In addition the data written might not be coherent with L3. This is because it is unknown whether the background operation has completed. In summary, there can still be dirty lines after a cache clean operation.

————— **Note** —————

Data accessed by the L1 master is still correct.

Software must not perform a clean instruction on a region when it contains active data, that is, data accessed during the clean operation. To ensure that a clean operation is completed, mask the interrupts. Also ensure that the software polls the Cache Operation Register to check if the operation is complete.

Table 3-16 shows the cache maintenance operations.

Table 3-16 Cache maintenance operations

Operation	Description
Cache Sync	Drain the STB. Operation complete when all buffers, LRB, LFB, STB, and EB, are empty.
Invalidate Line by PA	Specific L2 cache line is marked as not valid.
Invalidate by Way	Invalidate all data in specified ways, including dirty data. An Invalidate by way while selecting all cache ways is equivalent to invalidating all cache entries. Completes as a background task with the way(s) locked, preventing allocation.
Clean Line by PA	Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.
Clean Line by Index/Way	Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.
Clean by Way	Writes each line of the specified L2 cache ways to L3 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 way(s) locked, preventing allocation.
Clean and Invalidate Line by PA	Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid.
Clean and Invalidate Line by Index/Way	Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid.
Clean and Invalidate by Way	Writes each line of the specified L2 cache ways to L3 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 way(s) locked, preventing allocation.

During all operations where a cache line is cleaned or invalidated the non-secure bit is unchanged and is treated in the same way as the address.

System Cache Maintenance Considerations

This section introduces the detailed code sequences for cache maintenance for systems that include L1 and L2 caches. A Cortex-A9 processor connected to the L2CC (PL310) is a good example of this type of system. This section also describes the corner cases that can arise with *Multi Processor* (MP) L1 Caches and a system L2 cache (especially in exclusive cache configuration). It also provides a set of robust code sequences for cache maintenance for these cases.

The key architectural principles are:

- The controller can allocate a location with a valid mapping into the cache at any time. This can happen as a result of a simple read of a location. The read action can cause either a speculative prefetch, or a speculative loading in an MP system.
- The controller can evict a location in a cache at any time. New allocations into the cache can cause a cache eviction. Snoops from other agents coming into the Snoop Control Unit can also cause a cache eviction, see *Cortex™-A9 Technical Reference Manual*.

This section assumes the worst case condition that evictions and allocations occur at the most inconvenient times.

A working assumption is that the system has non-coherent components at Level 3 (typically for legacy reasons) with which software must be able to achieve coherency.

You can use the Clean and the Invalidate operations to achieve this coherency. Use the Clean operation to publish to external components any change in an ARM processor. Use the Invalidate operation to remove stale cache entries and make this visible externally.

Clean Operations and *Invalidate Operations* on page 3-31 describes how you can use Clean and Invalidate operations.

Clean Operations

This maintenance operation makes any change to the ARM cluster (including L1 and L2 caches) visible to external world. In this case the ARM system publishes new data for the external L3 system to use. This example assumes that there is not a race condition and therefore:

- the most recent version of the data is in the ARM cluster
- there are no new writes to the location, by other cores within the cluster that must be seen externally during the clean operation.

The pseudo code sequence for supporting this scenario is:

```
CleanLevel1 Address ; This is broadcast within the cluster
DSB                 ; Ensure completion of the clean as far as Level 2
CleanLevel2 Address ; forces the address out past level 2
CACHE SYNC          ; Ensures completion of the L2 clean
```

Invalidate Operations

This operation makes any change in the external L3 memory visible to the ARM cluster. The external system publishes new data for the ARM system to use. This example assumes that there is no race condition and therefore:

- the most recent version of the data is in the external memory cluster
- the location can be at any level in the ARM caches and is not dirty (you can do a clean operation of the ARM system before any external memory update).

For this scenario it looks as though you require a similar sequence as in *Clean Operations* on page 3-30, for example:

```
InvalLevel1 Address ; This is broadcast within the cluster
DSB                 ; Ensure completion of the inval as far as Level 2
InvalLevel2 Address ; forces the address out past level 2
CACHE SYNC          ; Ensures completion of the L2 inval.
```

But this sequence does not work robustly, because these examples assume that any line might be allocated into the cache at any time. If there is a stale entry in the L2 cache, the system enables the invalidation of the L1 cache. But before the controller invalidates the L2 cache, it allocates a line from the L2 cache to an L1 cache.

The robust code sequence for invalidation with a non-exclusive cache arrangement is:

```
InvalLevel2 Address ; forces the address out past level 2
SYNC                ; Ensures completion of the L2 inval
InvalLevel1 Address ; This is broadcast within the cluster
DSB                 ; Ensure completion of the inval as far as Level 2.
```

This sequence ensures that, if there is an allocation to L1 after the L1 invalidation, the data picked up is the new data and not stale data from the L2.

This sequence is not robust with an exclusive L2 cache, because an eviction of a clean line from L1 can be allocated to L2 in the period between the L2 invalidation and the L1 invalidation. So the L2 cache could contain the stale data with the L1 cache invalidated, therefore the invalidation sequence has failed.

Note

Repeated invalidations reduce the probability of an invalidation failure occurring. An example of a sequence in which two unlikely timed events would be necessary to cause an invalidation failure is:

1. InvalL2
2. InvalL1
3. InvalL2.

An example of a sequence that would make an invalidation failure even less probable is:

1. InvalL2
 2. InvalL1
 3. InvalL2
 4. InvalL1.
-

Clean and Invalidate Operations

Use this operation to:

- Shut down operating and disabled caches.

Note

You could also use the Clean operation followed by the Invalidate operation separately.

- Perform a combination of Clean and Invalidate operations that ensures coherency from the outside in, and from the inside out.

A Clean and Invalidate operation of both levels of cache behaves as a Clean operation followed by an Invalidate operation. Therefore the required sequence is:

```
CleanLevel1 Address ; This is broadcast within the cluster
DSB                 ; Ensure completion of the clean as far as Level 2
CleanLevel2 Address ; forces the address out past level 2
CACHE SYNC          ; Ensures completion of the L2 clean
InvalLevel2 Address ; forces the address out past level 2
CACHE SYNC          ; Ensures completion of the L2 inval
InvalLevel1 Address ; This is broadcast within the cluster
DSB                 ; Ensure completion of the inval as far as Level 2
```

The Clean and Invalidate operation enables this shortened sequence:

```
CleanLevel1 Address ; This is broadcast within the cluster
DSB                 ; Ensure completion of the clean as far as Level 2
Clean&InvalLevel2 Address ; forces the address out past level 2
```

```

CACHE SYNC                ; Ensures completion of the L2 inval
InvalLevel1 Address       ; This is broadcast within the cluster
DSB                       ; Ensure completion of the inval as far as Level 2

```

This sequence has the same problem with exclusive caches as *Invalidate Operations* on page 3-31. There is one additional issue, also with non-exclusive caches, if you use Clean and Invalidate operation as an alternative to the Clean operation. The issue is for when it is possible for one of the processors in the cluster to have a store during this sequence. This case constitutes a race condition. There is no guarantee that the system makes this data externally visible at the end of the sequence. The system expects the write not to be lost and so available for next time. This cannot happen if you use an Invalidate at L1. So the code sequence for Clean and Invalidate over the two levels of cache must be:

```

CleanLevel1 Address       ; This is broadcast within the cluster
DSB                       ; Ensure completion of the clean as far as Level 2
Clean&InvalLevel2 Address ; forces the address out past level 2
SYNC                     ; Ensures completion of the L2 inval
Clean&InvalLevel1 Address ; This is broadcast within the cluster
DSB                       ; Ensure completion of the clean&inval as far as Level 2 (no data lost)

```

Conclusion

This shows that with non-exclusive caches there are code sequences available for you to build the functionality of Clean, Invalidate and Clean, and Invalidate into a system that includes L1 and L2 caches. Take care when you write these sequences to avoid occasional sporadic failures in the two levels of cache.

The exclusive cache configuration between the L1 and L2 caches is more challenging and you must also take care when developing code to support cache maintenance sequences.

3.3.11 Cache Lockdown

These registers can prevent new addresses from being allocated and can also prevent data from being evicted out of the L2 cache. Such behavior can distinguish instructions from data transactions.

————— Note —————

Cache maintenance operations that invalidate, clean, or clean and invalidate cache contents affect locked-down cache lines as normal.

This register has read-only or read and write permission, depending on the security state you have selected for the register access and on the Non-Secure Lockdown Enable bit in the Auxiliary Control Register. Table 3-17 shows the different settings of the Cache Lockdown Register.

Table 3-17 Cache lockdown

Security of register access	Non-Secure Lockdown Enable bit	Permission
Secure	0, this is the default value	Read and write
	1	Read and write
Non-Secure	0, this is the default value	Read only
	1	Read and write

On reset the Non-Secure Lockdown Enable bit is set to 0 and Lockdown Registers are not permitted to be modified by non-secure accesses. In that configuration, if a non-secure access tries to write to those registers, the write response returns a DECERR response. This decode error results in the registers not being updated.

When permitted, the non-secure lockdown functionality can be identical to the secure one.

There are two lockdown schemes, line-based and way-based.

Cache lockdown by line

The following two registers enable the use of this optional lockdown by line feature:

- Lockdown by Line Enable Register. See Table 3-18 on page 3-35.
- Unlock All Lines Register. See Table 3-19 on page 3-35.

If you try to launch a background cache maintenance operation when the cache controller is performing an *unlock all lines* operation the controller returns SLVERR.

Table 3-18 Lockdown by Line Enable Register bit assignments

Bits	Field	Description
[31:1]	Reserved	SBZ/RAZ
[0]	lockdown_by_line_enable	0 = Lockdown by line disabled. This is the default. 1 = Lockdown by line disabled.

Table 3-19 Unlock All Lines Register bit assignments

Bits	Field	Description
[31:16]	Reserved	SBZ/RAZ
[15:0]	unlock_all_lines_by_way_operation	For all bits: 0 = Unlock all lines disabled. This is the default. 1 = Unlock all lines operation in progress for the corresponding way.

Cache lockdown by way

To control the *cache lockdown by way* and the *cache lockdown by master* mechanisms see the tables from Table 3-20 to Table 3-35 on page 3-38. See also *Lockdown by way* on page 2-23.

Table 3-20 Data Lockdown 0 Register, offset 0x900

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK000	Use when AR/WUSERSx[7:5] = 000

Table 3-21 Instruction Lockdown 0 Register, offset 0x904

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK000	Use when AR/WUSERSx[7:5] = 000

Table 3-22 Data Lockdown 1 Register, offset 0x908

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK001	Use when AR/WUSERSx[7:5] = 001

Table 3-23 Instruction Lockdown 1 Register, offset 0x90C

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK001	Use when AR/WUSERSx[7:5] = 001

Table 3-24 Data Lockdown 2 Register, offset 0x910

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK010	Use when AR/WUSERSx[7:5] = 010

Table 3-25 Instruction Lockdown 2 Register, offset 0x914

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK010	Use when AR/WUSERSx[7:5] = 010

Table 3-26 Data Lockdown 3 Register, offset 0x918

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK011	Use when AR/WUSERSx[7:5] = 011

Table 3-27 Instruction Lockdown 3 Register, offset 0x91C

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK011	Use when AR/WUSERSx[7:5] = 011

Table 3-28 Data Lockdown 4 Register, offset 0x920

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK100	Use when AR/WUSERSx [7:5] = 100

Table 3-29 Instruction Lockdown 4 Register, offset 0x924

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK100	Use when AR/WUSERSx [7:5] = 100

Table 3-30 Data Lockdown 5 Register, offset 0x928

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK101	Use when AR/WUSERSx [7:5] = 101

Table 3-31 Instruction Lockdown 5 Register, offset 0x92C

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK101	Use when AR/WUSERSx [7:5] = 101

Table 3-32 Data Lockdown 6 Register, offset 0x930

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK110	Use when AR/WUSERSx [7:5] = 110

Table 3-33 Instruction Lockdown 6 Register, offset 0x934

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK110	Use when AR/WUSERSx [7:5] = 110

Table 3-34 Data Lockdown 7 Register, offset 0x938

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK111	Use when AR/WUSERSx[7:5] = 111

Table 3-35 Instruction Lockdown 7 Register, offset 0x93C

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK111	Use when AR/WUSERSx[7:5] = 111

Note

- If the p1310_16_WAYS option is not implemented, bits [15:8] are reserved in all the Data and Instruction Lockdown registers.
- The Data and Instruction Lockdown 1-7 registers are not used if the option p1310_LOCKDOWN_BY_MASTER is not enabled. This corresponds to the simple Lockdown by Way, see *Lockdown by way* on page 2-23.

Replacement strategy

The cache controller uses a pseudo-random replacement strategy. A deterministic replacement strategy can be achieved when you use the lockdown registers. The pseudo-random replacement strategy fills invalid and unlocked ways first. For each line, when ways are either valid or locked, the victim is chosen as the next unlocked way.

If you require a deterministic replacement strategy, the lockdown registers are used to prevent ways from being allocated. For example, if the L2 size is 256KB, and each way is 32KB, and a piece of code is required to reside in two ways of 64KB, with a deterministic replacement strategy, then ways 1-7 must be locked before the code is filled into the L2 cache. If the first 32KB of code is allocated into way 0 only, then way 0 must be locked and way 1 unlocked so that the second half of the code can be allocated in way 1.

There are two lockdown registers, one for data and one for instructions. If required, you can separate data and instructions into separate ways of the L2 cache.

- Note

If p1310_LOCKDOWN_BY_MASTER is implemented, there are 16 lockdown registers.

3.3.12 Address Filtering

When two masters are implemented, you can redirect a whole address range to master 1 (M1).

When `address_filtering_enable` is set, all accesses with `address >= address_filtering_start` and `< address_filtering_end` are automatically directed to M1. All other accesses are directed to M0.

This feature is programmed using two registers.

Address Filtering Start Register

The Address Filtering Start Register is a read and write register. Figure 3-12 shows the register bit assignments.

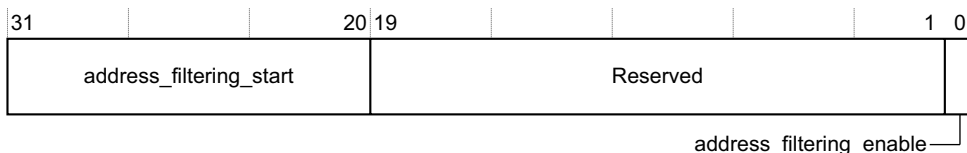


Figure 3-12 Address Filtering Start Register bit assignments

Table 3-36 shows the register bit assignments.

Table 3-36 Address Filtering Start Register bit assignments

Bits	Field	Description
[31:20]	address_filtering_start	Address filtering start address.
[19:1]	Reserved	SBZ/RAZ
[0]	address_filtering_enable	0 = Address filtering disabled. 1 = Address filtering enabled.

The value of the Address Filtering Start Register out of reset is given by the tied value of the **CFGADDRFILTSTART** and **CFGADDRFILTEN** input pins.

Note

It is recommended that you program the Address Filtering End Register before the Address Filtering Start Register to avoid unpredictable behavior between the two writes.

Address Filtering End Register

The Address Filtering End Register is a read and write register. Figure 3-13 shows the register bit assignments.



Figure 3-13 Address Filtering End Register bit assignments

Table 3-37 shows the register bit assignments.

Table 3-37 Address Filtering End Register bit assignments

Bits	Field	Description
[31:20]	address_filtering_end	Address filtering end address
[19:0]	Reserved	SBZ/RAZ

The value of the Address Filtering End Register out of reset is given by the tied value of the **CFGADDRFILTEND** input pin.

3.3.13 Debug Register

The Debug Control Register forces specific cache behavior required for debug. This register has read-only (non-secure) or read/write (secure) permission. Any secure access and non-secure access can read this register. Only a secure access can write to this register. If a non-secure access tries to write to this register the register issues a DECERR response and does not update.

Figure 3-14 on page 3-41 shows the register bit assignments.

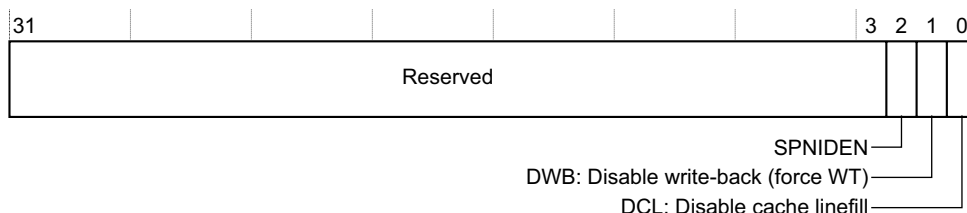


Figure 3-14 Debug Control Register bit assignments

Table 3-38 shows the register bit assignments.

Table 3-38 Debug Control Register bit assignments

Bits	Field	Description
[31:3]	Reserved	SBZ/RAZ
[2]	SPNIDEN	Reads value of SPNIDEN input.
[1]	DWB: Disable write-back (force WT)	0 = Enable write-back behavior. This is the default. 1 = Force write-through behavior.
[0]	DCL: Disable cache linefill	0 = Enable cache linefills. This is the default. 1 = Disable cache linefills.

Forcing write-through behavior

If you set the DWB bit to 1, it forces the cache controller to treat all cacheable writes as though they were in a write-through no write-allocate region of memory. The setting of the DWB bit overrides the 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.

While the DWB bit is set, lines that are clean are not marked as dirty if they are updated. 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 you have set the DWB bit to 1, 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.

Disabling cache linefills

If you set the DCL bit to 1, no allocation occurs on either reads or writes. This mode of operation is required for debug so that the memory image, as seen by the processor, can be examined in a non-invasive manner. Cache hits read data words from the cache, and cache misses from a cacheable region read words directly from memory.

Note

The forcing write-through and disabling cache linefills features have priority over other features acting on cachability properties, such as Force Write-Allocate and exclusive cache configuration.

3.3.14 Prefetch Offset Register

You can enable prefetch by setting bit 29 or 28 of the Auxiliary Control Register. If you enable prefetch and one of the slave ports receives a cacheable read transaction, a cache lookup is done on a following cache line. The address of the cache line is given by bits [4:0] of the Prefetch Offset Register. In case of miss, the cache line is prefetched from L3 and allocated into the L2 cache.

By default, the prefetch offset is equal to 5'b00000. This means that the following cache line is prefetched. For example, if S0 receives a cacheable read at address 0x100, the cache line at address 0x120 is prefetched. Prefetching the following cache line might not give optimal performance. In some systems, it might be better to prefetch more in advance to achieve better performance. The prefetch offset allows this by setting the address of the prefetched cache line to Cache Line + 1 + Offset. The optimal value of the prefetch offset depends on the L3 read latency and on the L1 read issuing capability. It is recommended to do system experiments by varying the prefetch offset in order to find the optimal value.

An example is when PL310 is connected to a Cortex-A9 uniprocessor. Experiments, made by running a memcpy routine, show that a prefetch offset programmed with the value 0x06 (5'b00110) permit the best performance, no matter what the L3 latency.

Figure 3-15 shows the register bit assignments.

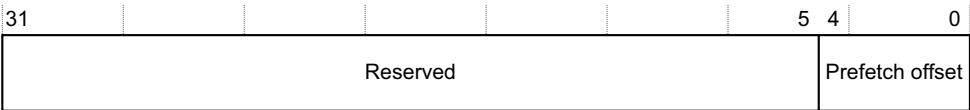


Figure 3-15 Prefetch Offset Register bit assignments

Table 3-39 shows the register bit assignments.

Table 3-39 Prefetch Offset Register bit assignments

Bits	Field	Description
[31:5]	Reserved	SBZ/RAZ
[4:0]	Prefetch offset	Default = 00000.

The prefetch mechanism is not launched in the case of 4KB boundary crossing.

Appendix A

Signal Descriptions

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

- *Clock and reset* on page A-2
- *Configuration* on page A-3
- *Slave and master ports* on page A-4
- *RAM interface* on page A-12
- *Cache event monitoring* on page A-15
- *Cache interrupt* on page A-16
- *MBIST interface* on page A-17.

A.1 Clock and reset

Table A-1 shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Type	Description
CLK	Input	Main clock
CLKOUT	Output	Global clock signal for RAMs
DATACLKEN	Input	Clock enable for Data RAM interface
DATACLKOUT	Output	Clock for Data RAM
DATACLKOUTEN	Output	Clock enable for Data RAM clock
IDLE	Output	Indicates cache controller is idle
INCLKENM0	Input	Clock enable for M0 AXI inputs
INCLKENM1	Input	Clock enable for M1 AXI inputs
INCLKENS0	Input	Clock enable for S0 AXI inputs
INCLKENS1	Input	Clock enable for S1 AXI inputs
nRESET	Input	Global reset, active LOW
OUTCLKENM0	Input	Clock enable for M0 AXI outputs
OUTCLKENM1	Input	Clock enable for M1 AXI outputs
OUTCLKENS0	Input	Clock enable for S0 AXI outputs
OUTCLKENS1	Input	Clock enable for S1 AXI outputs
TAGCLKEN	Input	Clock enable for Tag RAM interface
TAGCLKOUT	Output	Clock for Tag RAM
TAGCLKOUTEN	Output	Clock enable for Tag RAM clock

A.2 Configuration

Table A-2 shows the configuration signals.

Table A-2 Configuration signals

Signal	Type	Description
ASSOCIATIVITY	Input	Associativity for Auxiliary Control Register. See <i>Auxiliary Control Register</i> on page 3-12.
CACHEID[5:0]	Input	Cache controller cache ID
CFGADDRFILTE ^a	Input	Address filtering Enable out of reset
CFGADDRFILTE ^a	Input	Address filtering End Address out of reset
CFGADDRFILTE ^a	Input	Address filtering Start Address out of reset
CFGBIGEND	Input	Big-endian mode for accessing configuration registers out of reset
DATAREADLAT[2:0]	Output	Read access latency for Data RAM
DATASETUPLAT[2:0]	Output	Setup latency for Data RAM
DATAWRITELAT[2:0]	Output	Write access latency for Data RAM
REGFILEBASE[19:0]	Input	Base address for accessing configuration registers
SE	Input	DFT test enable, held HIGH during serial shift of scan chains and LOW for capture
SPNIDEN	Input	Secure privileged non-invasive debug enable
TAGREADLAT[2:0]	Output	Read access latency for Tag RAM
TAGSETUPLAT[2:0]	Output	Setup latency for Tag RAM
TAGWRITELAT[2:0]	Output	Write access latency for Tag RAM
WAYSIZ[2:0]	Input	Size of ways for Auxiliary Control Register. See <i>Auxiliary Control Register</i> on page 3-12

a. For address filtering implementation.

A.3 Slave and master ports

The slave and master ports are described in the following sections:

- *Slave port 0*
- *Slave port 1* on page A-6
- *Master port 0* on page A-8
- *Master port 1* on page A-10.

A.3.1 Slave port 0

Table A-3 shows the slave port 0 signals.

Table A-3 Slave port 0 signals

Signal	Type	Description
ARADDRS0[31:0]	Input	Address bus
ARBURSTS0[1:0]	Input	Burst type
ARCACHES0[3:0]	Input	Cache information
ARIDS0[[~] p1310_AXI_ID_MAX:0]	Input	Address ID
ARLENS0[3:0]	Input	Burst length
ARLOCKS0[1:0]	Input	Lock type
ARPROTS0[2:0]	Input	Protection information
ARREADYS0	Output	Address accepted
ARSIZES0[2:0]	Input	Burst size
ARUSERS0[9:8]	Input	Hint signals from ARM Cortex-A9 processor
ARUSERS0[7:5]	Input	Master ID for lockdown by master
ARUSERS0[4:1]	Input	Inner cacheable attributes
ARUSERS0[0]	Input	Shared attribute
ARVALIDS0	Input	Address valid
AWADDRS0[31:0]	Input	Address bus
AWBURSTS0[1:0]	Input	Burst type
AWCACHES0[3:0]	Input	Cache information

Table A-3 Slave port 0 signals (continued)

Signal	Type	Description
AWIDS0[$\text{pl310_AXI_ID_MAX:0}$]	Input	Address ID
AWLENS0[3:0]	Input	Burst length
AWLOCKS0[1:0]	Input	Lock type
AWPROTS0[2:0]	Input	Protection information
AWREADYS0	Output	Address accepted
AWSIZES0[2:0]	Input	Burst size
AWUSERS0[11:10]	Input	Hint signals from ARM Cortex-A9 processor
AWUSERS0[9]	Input	Indicates a clean eviction ^a
AWUSERS0[8]	Input	Indicates an eviction ^a
AWUSERS0[7:5]	Input	Master ID for lockdown by master
AWUSERS0[4:1]	Input	Inner cacheable attributes
AWUSERS0[0]	Input	Shared attribute
AWVALIDS0	Input	Address valid
BIDS0[$\text{pl310_AXI_ID_MAX:0}$]	Output	Write ID
BREADYS0	Input	Write response accepted
BRESPS0[1:0]	Output	Write response
BVALIDS0	Output	Write response valid
RDATAS0[63:0]	Output	Read data bus
RIDS0[$\text{pl310_AXI_ID_MAX:0}$]	Output	Read ID
RLASTS0	Output	Read last transfer
RREADYS0	Input	Read accepted
RRESPS0[1:0]	Output	Read response
RVALIDS0	Output	Read data valid
WDATAS0[63:0]	Input	Write data bus
WIDS0[$\text{pl310_AXI_ID_MAX:0}$]	Input	Write ID

Table A-3 Slave port 0 signals (continued)

Signal	Type	Description
WLASTS0	Input	Write last transfer
WREADY0	Output	Write data accepted
WSTRBS0[7:0]	Input	Write strobes
WVALID0	Input	Write data valid

a. Exclusive cache configuration only.

A.3.2 Slave port 1

Slave port 1 is only implemented in a two-slave configuration. Table A-4 shows the slave port 1 signals.

Table A-4 Slave port 1 signals

Signal	Type	Description
ARADDRS1[31:0]	Input	Address bus
ARBURSTS1[1:0]	Input	Burst type
ARCACHES1[3:0]	Input	Cache information
ARIDS1[\negpl310_AXI_ID_MAX:0]	Input	Address ID
ARLENS1[3:0]	Input	Burst length
ARLOCKS1[1:0]	Input	Lock type
ARPROTS1[2:0]	Input	Protection information
ARREADY1	Output	Address accepted
ARSIZES1[2:0]	Input	Burst size
ARUSERS1[9:8]	Input	Hint signals from ARM Cortex-A9 processor
ARUSERS1[7:5]	Input	Master ID for lockdown by master
ARUSERS1[4:1]	Input	Inner cacheable attributes
ARUSERS1[0]	Input	Shared attribute
ARVALID1	Input	Address valid

Table A-4 Slave port 1 signals (continued)

Signal	Type	Description
AWADDRS1[31:0]	Input	Address bus
AWBURSTS1[1:0]	Input	Burst type
AWCACHES1[3:0]	Input	Cache information
AWIDS1[\negpl310_AXI_ID_MAX:0]	Input	Address ID
AWLENS1[3:0]	Input	Burst length
AWLOCKS1[1:0]	Input	Lock type
AWPROTS1[2:0]	Input	Protection information
AWREADYS1	Output	Address accepted
AWSIZES1[2:0]	Input	Burst size
AWUSERS1[11:10]	Input	Hint signals from ARM Cortex-A9 processor
AWUSERS1[9]	Input	Indicates a clean eviction ^a
AWUSERS1[8]	Input	Indicates an eviction ^a
AWUSERS1[7:5]	Input	Master ID for lockdown by master
AWUSERS1[4:1]	Input	Inner cacheable attributes
AWUSERS1[0]	Input	Shared attribute
AWVALIDS1	Input	Address valid
BIDS1[\negpl310_AXI_ID_MAX:0]	Output	Write ID
BREADYS1	Input	Write response accepted
BRESPS1[1:0]	Output	Write response
BVALIDS1	Output	Write response valid
RDATAS1[63:0]	Output	Read data bus
RIDS1[\negpl310_AXI_ID_MAX:0]	Output	Read ID
RLASTS1	Output	Read last transfer
RREADYS1	Input	Read accepted
RRESPS1[1:0]	Output	Read response

Table A-4 Slave port 1 signals (continued)

Signal	Type	Description
RVALIDS1	Output	Read data valid
WDATAS1[63:0]	Input	Write data bus
WIDS1[pl310_AXI_ID_MAX:0]	Input	Write ID
WLASTS1	Input	Write last transfer
WREADYS1	Output	Write data accepted
WSTRBS1[7:0]	Input	Write strobes
WVALIDS1	Input	Write data valid

a. Exclusive cache configuration only.

A.3.3 Master port 0

Table A-5 shows the master port 0 signals.

Table A-5 Master port 0 signals

Signal	Type	Description
ARADDRM0[31:0]	Output	Address bus
ARBURSTM0[1:0]	Output	Burst type
ARCACHEM0[3:0]	Output	Cache information
ARIDM0[pl310_AXI_ID_MAX+2:0]	Output	Address ID
ARLENM0[3:0]	Output	Burst length
ARLOCKM0[1:0]	Output	Lock type
ARPROTM0[2:0]	Output	Protection information
ARREADYM0	Input	Address accepted
ARSIEM0[2:0]	Output	Burst size
ARVALIDM0	Output	Address valid
AWADDRM0[31:0]	Output	Address bus
AWBURSTM0[1:0]	Output	Burst type

Table A-5 Master port 0 signals (continued)

Signal	Type	Description
AWCACHEM0[3:0]	Output	Cache information
AWIDM0[\negpl310_AXI_ID_MAX+2:0]	Output	Address ID
AWLENM0[3:0]	Output	Burst length
AWLOCKM0[1:0]	Output	Lock type
AWPROTM0[2:0]	Output	Protection information
AWREADYM0	Input	Address accepted
AWSIZEM0[2:0]	Output	Burst size
AWVALIDM0	Output	Address valid
BIDM0[\negpl310_AXI_ID_MAX+2:0]	Input	Write ID
BREADYM0	Output	Write response accepted
BRESPM0[1:0]	Input	Write response
BVALIDM0	Input	Write response valid
RDATAM0[63:0]	Input	Read data bus
RIDM0[\negpl310_AXI_ID_MAX+2:0]	Input	Read ID
RLASTM0	Input	Read last transfer
RREADYM0	Output	Read accepted
RRESPM0[1:0]	Input	Read response
RVALIDM0	Input	Read data valid
WDATAM0[63:0]	Output	Write data bus
WIDM0[\negpl310_AXI_ID_MAX+2:0]	Output	Write ID
WLASTM0	Output	Write last transfer
WREADYM0	Input	Write data accepted
WSTRBM0[7:0]	Output	Write strobes
WVALIDM0	Output	Write data valid

A.3.4 Master port 1

Master port 1 is only implemented in a two-master configuration. Table A-6 shows the master port 1 signals.

Table A-6 Master port 1 signals

Signal	Type	Description
ARADDRM1[31:0]	Output	Address bus
ARBURSTM1[1:0]	Output	Burst type
ARCACHEM1[3:0]	Output	Cache information
ARIDM1[$\text{pl310_AXI_ID_MAX}+2:0$]	Output	Address ID
ARLENM1[3:0]	Output	Burst length
ARLOCKM1[1:0]	Output	Lock type
ARPROTM1[2:0]	Output	Protection information
ARREADYM1	Input	Address accepted
ARSIZEM1[2:0]	Output	Burst size
ARVALIDM1	Output	Address valid
AWADDRM1[31:0]	Output	Address bus
AWBURSTM1[1:0]	Output	Burst type
AWCACHEM1[3:0]	Output	Cache information
AWIDM1[$\text{pl310_AXI_ID_MAX}+2:0$]	Output	Address ID
AWLENM1[3:0]	Output	Burst length
AWLOCKM1[1:0]	Output	Lock type
AWPROTM1[2:0]	Output	Protection information
AWREADYM1	Input	Address accepted
AWSIZEM1[2:0]	Output	Burst size
AWVALIDM1	Output	Address valid
BIDM1[$\text{pl310_AXI_ID_MAX}+2:0$]	Input	Write ID
BREADYM1	Output	Write response accepted

Table A-6 Master port 1 signals (continued)

Signal	Type	Description
BRESPM1[1:0]	Input	Write response
BVALIDM1	Input	Write response valid
RDATAM1[63:0]	Input	Read data bus
RIDM1[\negpl310_AXI_ID_MAX+2:0]	Input	Read ID
RLASTM1	Input	Read last transfer
RREADYM1	Output	Read accepted
RRESPM1[1:0]	Input	Read response
RVALIDM1	Input	Read data valid
WDATAM1[63:0]	Output	Write data bus
WIDM1[\negpl310_AXI_ID_MAX+2:0]	Output	Write ID
WLASTM1	Output	Write last transfer
WREADYM1	Input	Write data accepted
WSTRBM1[7:0]	Output	Write strobes
WVALIDM1	Output	Write data valid

A.4 RAM interface

The RAM interface consists of these sub interfaces:

- Data RAM interface
- Tag RAM interface on page A-13.

A.4.1 Data RAM interface

Table A-7 shows the data RAM interface signals.

Table A-7 Data RAM interface signals

Signal	Type	Description
DATAADDR[17:0] ^a DATAADDR[16:0] ^b	Output	Data RAM address
DATACS	Output	Data RAM chip select
DATAEN[31:0]	Output	Data RAM byte write enables
DATAERR	Input	Data RAM error
DATA _n RW	Output	Data RAM write control signal
DATAPRD[31:0] ^c	Input	Data RAM parity read data
DATAPWD[31:0] ^c	Output	Data RAM parity write data
DATARD[255:0]	Input	Data RAM read data
DATAWAIT	Input	Data RAM wait
DATAWD[255:0]	Output	Data RAM write data

a. For 16-way implementation.
b. For 8-way implementation.
c. Optional. Only present if p1310_PARITY is defined.

A.4.2 Tag RAM interface

Table A-8 shows the tag RAM interface signals.

Table A-8 Tag RAM interface

Signal	Type	Description
TAGADDR[13:0]	Output	Tag RAM address
TAGCS[15:0]^a TAGCS[7:0]^b	Output	Tag RAM chip selects
TAGEN[20:0]	Output	Tag RAM write enable
TAGLEN^c	Output	Tag RAM lock write enable
TAGERR[15:0]^a TAGERR[7:0]^b	Input	Tag RAM error
TAGnRW	Output	Tag RAM write control
TAGPRD[15:0]^{ad} TAGPRD[7:0]^{bd}	Input	Tag RAM parity read data
TAGLRD[15:0]^{ac} TAGLRD[7:0]^{bc}	Input	Tag RAM lock read data
TAGPEN^d	Output	Tag RAM parity write enable
TAGPWD^d	Output	Tag RAM parity write data
TAGLWD^c	Output	Tag RAM lock write data
TAGRD0[20:0]	Input	Tag RAM 0 read data
TAGRD1[20:0]	Input	Tag RAM 1 read data
TAGRD2[20:0]	Input	Tag RAM 2 read data
TAGRD3[20:0]	Input	Tag RAM 3 read data
TAGRD4[20:0]	Input	Tag RAM 4 read data
TAGRD5[20:0]	Input	Tag RAM 5 read data
TAGRD6[20:0]	Input	Tag RAM 6 read data
TAGRD7[20:0]	Input	Tag RAM 7 read data
TAGRD8[20:0]^e	Input	Tag RAM 8 read data

Table A-8 Tag RAM interface (continued)

Signal	Type	Description
TAGRD9[20:0]^e	Input	Tag RAM 9 read data
TAGRD10[20:0]^e	Input	Tag RAM 10 read data
TAGRD11[20:0]^e	Input	Tag RAM 11 read data
TAGRD12[20:0]^e	Input	Tag RAM 12 read data
TAGRD13[20:0]^e	Input	Tag RAM 13 read data
TAGRD14[20:0]^e	Input	Tag RAM 14 read data
TAGRD15[20:0]^e	Input	Tag RAM 15 read data
TAGWAIT	Input	Tag RAM wait
TAGWD[20:0]	Output	Tag RAM write data

- a. For 16-way implementation.
- b. For 8-way implementation.
- c. Optional. Only present if p1310_LOCKDOWN_BY_LINE is defined.
- d. Optional. Only present if p1310_PARITY is defined.
- e. Only present for 16-way implementation.

A.5 Cache event monitoring

Table A-9 shows the cache event monitoring signals. See also *Cache event monitoring* on page 2-44.

Table A-9 Cache event monitoring signals

Signal	Type	Description
CO	Output	Eviction (CastOut) of a line from the L2 cache
DRHIT	Output	Data read hit
DRREQ	Output	Data read request
DWHIT	Output	Data write hit
DWREQ	Output	Data write request
DWTREQ	Output	Data write request with write-through attribute
IRHIT	Output	Instruction read hit
IRREQ	Output	Instruction read request
PF	Output	Prefetch linefill sent to L3
WA	Output	Write allocate

A.6 Cache interrupt

Table A-10 shows the cache interrupt signals.

Table A-10 Cache Interrupt signals

Signal	Type	Description
DECERRINTR	Output	Decode error received on master port from L3
ECNTRINTR	Output	Event Counter Overflow/Increment
ERRRDINTR	Output	Error on L2 data RAM read
ERRRTINTR	Output	Error on L2 tag RAM read
ERRWDINTR	Output	Error on L2 data RAM write
ERRWTINTR	Output	Error on L2 data RAM write
L2CCINTR	Output	Combined Interrupt Output
PARRDINTR	Output	Parity error on L2 data RAM read
PARRTINTR	Output	Parity error on L2 tag RAM read
SLVERRINTR	Output	Slave error received on master port from L3

A.7 MBIST interface

Table A-11 shows the MBIST interface signals.

Table A-11 MBIST interface signals

Signal	Type	Description
MBISTADDR[19:0]	Input	MBIST address
MBISTCE[17:0]	Input	MBIST RAM chip enable
MBISTDCTL[19:0]	Output	MBIST data out multiplexor control
MBISTDIN[63:0]	Input	MBIST data in
MBISTWE[31:0]	Input	MBIST write enable
MBISTON	Input	MBIST mode enable
MBISTDOUT[63:0]	Output	MBIST data out

Appendix B

AC Parameters

This appendix specifies the AC timing requirements. All minimum timing parameters have the value of clock uncertainty. The maximum timing parameters or constraint for each cache controller signal applied to the SoC is provided as a percentage in the tables in this chapter. This appendix contains the following sections:

- *Reset and configuration* on page B-2
- *Slave port 0 I/O* on page B-3
- *Slave port 1 I/O* on page B-5
- *Master port 0 I/O* on page B-7
- *Master port 1 I/O* on page B-9
- *Data RAM* on page B-11
- *Event monitor* on page B-14
- *Cache interrupt ports* on page B-15
- *MBIST interface signal* on page B-16.

Note

This appendix specifies the AC timing requirements used during the development of the cache controller. The timing values listed are for information only.

B.1 Reset and configuration signal timing parameters

Table B-1 shows the reset and configuration signal timing parameters.

Table B-1 Reset and configuration		
Port name	I/O type	Maximum constraint
ASSOCIATIVITY[3:0]	Input	70%
CACHEID[5:0]	Input	20%
CFGADDRFILTEN	Input	70%
CFGADDRFILTEND[11:0]	Input	70%
CFGADDRFILTSTART[11:0]	Input	70%
CFGBIGEND	Input	70%
DATAREADLAT[2:0]	Output	70%
DATASETUPLAT[2:0]	Output	70%
DATAWRITELAT[2:0]	Output	70%
nRESETn	Input	20%
SE	Input	20%
TAGREADLAT[2:0]	Output	70%
TAGSETUPLAT[2:0]	Output	70%
TAGWRITELAT[2:0]	Output	70%
WAYSIZE[2:0]	Input	70%

B.2 Slave port 0 I/O signal timing parameters

Table B-2 shows the slave port 0 I/O signal timing parameters.

Table B-2 Slave port 0 I/O

Port name	I/O type	Maximum constraint
ARADDRS0[31:0]	Input	50%
ARBURSTS0[1:0]	Input	70%
ARCACHES0[3:0]	Input	50%
ARIDS0[5:0]	Input	50%
ARLENS0[3:0]	Input	70%
ARLOCKS0[1:0]	Input	50%
ARPROTS0[2:0]	Input	70%
ARREADYS0	Output	70%
ARSIZES0[2:0]	Input	70%
ARUSERS0[9:0]	Input	50%
ARVALIDS0	Input	50%
AWADDRS0[31:0]	Input	50%
AWBURSTS0[1:0]	Input	70%
AWCACHES0[3:0]	Input	70%
AWIDS0[5:0]	Input	70%
AWLENS0[3:0]	Input	70%
AWLOCKS0[1:0]	Input	50%
AWPROTS0[2:0]	Input	70%
AWREADYS0	Output	70%
AWSIZES0[2:0]	Input	70%
AWUSERS0[11:0]	Input	70%
AWVALIDS0	Input	50%
BIDS0[5:0]	Output	70%

Table B-2 Slave port 0 I/O (continued)

Port name	I/O type	Maximum constraint
BREADYS0	Input	60%
BRESPS0[1:0]	Output	70%
BVALIDS0	Output	70%
INCLKENS0	Input	30%
OUTCLKENS0	Output	30%
RDATAS0[63:0]	Output	70%
RIDS0[5:0]	Output	70%
RLASTS0	Output	70%
RREADYS0	Input	50%
RRESPS0[1:0]	Output	50%
RVALIDS0	Output	70%
WDATAS0[63:0]	Input	70%
WIDS0[5:0]	Input	70%
WLASTS0	Input	50%
WREADYS0	Output	70%
WSTRBS0[7:0]	Input	70%
WVALIDS0	Input	50%

B.3 Slave port 1 I/O signal timing parameters

Table B-3 shows the slave port 1 I/O signal timing parameters.

Table B-3 Slave port 1 I/O

Port name	I/O type	Maximum constraint
ARADDRS1[31:0]	Input	50%
ARBURSTS1[1:0]	Input	70%
ARCACHES1[3:0]	Input	50%
ARIDS1[5:0]	Input	50%
ARLENS1[3:0]	Input	70%
ARLOCKS1[1:0]	Input	50%
ARPROTS1[2:0]	Input	70%
ARREADYS1	Output	70%
ARSIZES1[2:0]	Input	70%
ARUSERS1[9:0]	Input	50%
ARVALIDS1	Input	50%
AWADDRS1[31:0]	Input	50%
AWBURSTS1[1:0]	Input	70%
AWCACHES1[3:0]	Input	70%
AWIDS1[5:0]	Input	70%
AWLENS1[3:0]	Input	70%
AWLOCKS1[1:0]	Input	50%
AWPROTS1[2:0]	Input	70%
AWREADYS1	Output	70%
AWSIZES1[2:0]	Input	70%
AWUSERS1[11:0]	Input	70%
AWVALIDS1	Input	50%
BIDS1[5:0]	Output	70%

Table B-3 Slave port 1 I/O (continued)

Port name	I/O type	Maximum constraint
BREADYS1	Input	60%
BRESPS1[1:0]	Output	70%
BVALIDS1	Output	70%
INCLKENS1	Input	30%
OUTCLKENS1	Output	30%
RDATAS1[63:0]	Output	70%
RIDS1[5:0]	Output	70%
RLASTS1	Output	70%
RREADYS1	Input	50%
RRESPS1[1:0]	Output	50%
RVALIDS1	Output	70%
WDATAS1[63:0]	Input	70%
WIDS1[5:0]	Input	70%
WLASTS1	Input	50%
WREADYS1	Output	70%
WSTRBS1[7:0]	Input	70%
WVALIDS1	Input	50%

B.4 Master port 0 I/O signal timing parameters

Table B-4 shows the master port 0 I/O signal timing parameters.

Table B-4 Master port 0 I/O

Port name	I/O type	Maximum constraint
ARADDRM0[31:0]	Output	70%
ARBURSTM0[1:0]	Output	70%
ARCACHEM0[3:0]	Output	70%
ARIDM0[7:0]	Output	70%
ARLENM0[3:0]	Output	70%
ARLOCKM0[1:0]	Output	70%
ARPROTM0[2:0]	Output	70%
ARREADYM0	Input	50%
ARSIZEM0[2:0]	Output	70%
ARVALIDM0	Output	70%
AWADDRM0[31:0]	Output	70%
AWBURSTM0[1:0]	Output	70%
AWCACHEM0[3:0]	Output	70%
AWIDM0[7:0]	Output	70%
AWLENM0[3:0]	Output	70%
AWLOCKM0[1:0]	Output	70%
AWPROTM0[2:0]	Output	70%
AWREADYM0	Input	50%
AWSIZEM0[2:0]	Output	70%
AWVALIDM0	Output	70%
BIDM0[7:0]	Input	50%
BREADYM0	Output	70%
BRESPM0[1:0]	Input	50%

Table B-4 Master port 0 I/O (continued)

Port name	I/O type	Maximum constraint
BVALIDM0	Input	50%
INCLKENM0	Input	30%
OUTCLKENM0	Output	30%
RDATAM0[63:0]	Input	50%
RIDM0[7:0]	Input	50%
RLASTM0	Input	50%
RREADYM0	Output	70%
RRESPM0[1:0]	Input	50%
RVALIDM0	Input	50%
WDATAM0[63:0]	Output	70%
WIDM0[7:0]	Output	70%
WLASTM0	Output	70%
WREADYM0	Input	50%
WSTRBM0[7:0]	Output	70%
WVALIDM0	Output	70%

B.5 Master port 1 I/O signal timing parameters

Table B-5 shows the master port 1 I/O signal timing parameters.

Table B-5 Master port 1 I/O

Port name	I/O type	Maximum constraint
ARADDRM1[31:0]	Output	70%
ARBURSTM1[1:0]	Output	70%
ARCACHEM1[3:0]	Output	70%
ARIDM1[7:0]	Output	70%
ARLENM1[3:0]	Output	70%
ARLOCKM1[1:0]	Output	70%
ARPROTM1[2:0]	Output	70%
ARREADYM1	Input	50%
ARSIZEM1[2:0]	Output	70%
ARVALIDM1	Output	70%
AWADDRM1[31:0]	Output	70%
AWBURSTM1[1:0]	Output	70%
AWCACHEM1[3:0]	Output	70%
AWIDM1[7:0]	Output	70%
AWLENM1[3:0]	Output	70%
AWLOCKM1[1:0]	Output	70%
AWPROTM1[2:0]	Output	70%
AWREADYM1	Input	50%
AWSIZEM1[2:0]	Output	70%
AWVALIDM1	Output	70%
BIDM1[7:0]	Input	50%
BREADYM1	Output	70%
BRESPM1[1:0]	Input	50%

Table B-5 Master port 1 I/O (continued)

Port name	I/O type	Maximum constraint
BVALIDM1	Input	50%
INCLKENM1	Input	30%
OUTCLKENM1	Output	30%
RDATAM1[63:0]	Input	50%
RIDM1[7:0]	Input	50%
RLASTM1	Input	50%
RREADYM1	Output	70%
RRESPM1[1:0]	Input	50%
RVALIDM1	Input	50%
WDATAM1[63:0]	Output	70%
WIDM1[7:0]	Output	70%
WLASTM1	Output	70%
WREADYM1	Input	50%
WSTRBM1[7:0]	Output	70%
WVALIDM1	Output	70%

B.6 RAMs signal timing parameters

This section shows the RAMs signal timing parameters in:

- *Data RAM*
- *Tag RAM* on page B-12.

B.6.1 Data RAM

Table B-6 shows the Data RAM signal timing parameters.

Table B-6 Data RAM

Port name	I/O type	Maximum constraint
DATAADDR[16:0]	Output	70%
DATACLKEN	Input	30%
DATACLKOUT	Output	50%
DATACLKOUTEN	Output	50%
DATAACS	Output	70%
DATAEN[31:0]	Output	70%
DATAERR	Input	50%
DATAnRW	Output	70%
DATAPEN[31:0]	Output	70%
DATAPnRW	Output	70%
DATAPRD[31:0]	Input	50%
DATAPWD[31:0]	Output	70%
DATARD[255:0]	Input	50%
DATAWD[255:0]	Output	70%
DATAWAIT	Input	30%

B.6.2 Tag RAM

Table B-7 shows the Tag RAM signal timing parameters.

Table B-7 Tag RAM

Port name	I/O type	Maximum constraint
TAGADDR[12:0]	Output	70%
TAGCLKEN	Input	30%
TAGCLKOUT	Output	50%
TAGCLKOUTEN	Output	50%
TAGCS[15:0]	Output	70%
TAGEN[20:0]	Output	70%
TAGERR[7:0]	Input	70%
TAGLEN	Output	70%
TAGLRD[15:0]	Input	70%
TAGLWD	Output	70%
TAGPRD[7:0]	Input	70%
TAGPWD	Output	70%
TAGRD0[20:0]	Input	70%
TAGRD1[20:0]	Input	70%
TAGRD10[20:0]	Input	70%
TAGRD11[20:0]	Input	70%
TAGRD12[20:0]	Input	70%
TAGRD13[20:0]	Input	70%
TAGRD14[20:0]	Input	70%
TAGRD15[20:0]	Input	70%
TAGRD2[20:0]	Input	70%
TAGRD3[20:0]	Input	70%
TAGRD4[20:0]	Input	70%

Table B-7 Tag RAM (continued)

Port name	I/O type	Maximum constraint
TAGRD5[20:0]	Input	70%
TAGRD6[20:0]	Input	70%
TAGRD7[20:0]	Input	70%
TAGRD8[20:0]	Input	70%
TAGRD9[20:0]	Input	70%
TAGWD[19:0]	Output	70%
TAGWAIT	Input	50%

B.7 Event monitor signal timing parameters

Table B-8 shows the event monitor signal timing parameters.

Table B-8 Event monitor		
Port name	I/O type	Maximum constraint
CO	Output	70%
DRHIT	Output	70%
DRREQ	Output	70%
DWHIT	Output	70%
DWREQ	Output	70%
DWTREQ	Output	70%
IRHIT	Output	70%
IRREQ	Output	70%
SPNIDEN	Input	70%
WA	Output	70%

B.8 Cache interrupt ports signal timing parameters

Table B-9 shows the cache interrupt ports signal timing parameters.

Table B-9 Cache interrupt ports

Port name	I/O type	Maximum constraint
DECERRINTR	Output	70%
ECNTRINTR	Output	70%
ERRRDINTR	Output	70%
ERRRTINTR	Output	70%
ERRWDINTR	Output	70%
ERRWTINTR	Output	70%
L2CCINTR	Output	70%
PARRDINTR	Output	70%
PARRTINTR	Output	70%
SLVERRINTR	Output	70%

B.9 MBIST interface signal timing parameters

Table B-10 shows the MBIST interface signal timing parameters.

Table B-10 MBIST interface signal		
Port name	I/O type	Maximum constraint
MBISTADDR[18:0]	Input	70%
MBISTCE[17:0]	Input	70%
MBISTDCTL[19:0]	Output	70%
MBISTDIN[63:0]	Input	70%
MBISTWE	Input	70%
MBISTON	Input	30%
MBISTDOUT[63:0]	Output	50%

Appendix C

Timing Diagrams

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

- *Single read hit transaction* on page C-2
- *Single read miss transaction* on page C-3
- *Single noncacheable read transaction* on page C-4
- *Outstanding read hit transaction* on page C-5
- *Hit under miss read transactions* on page C-6
- *Single bufferable write transaction* on page C-7
- *Single nonbufferable write transaction* on page C-8.

Note

No latency on RAMs is assumed in the timing diagrams in this appendix.

C.1 Single read hit transaction

Figure C-1 shows the timing for a single read hit transaction.

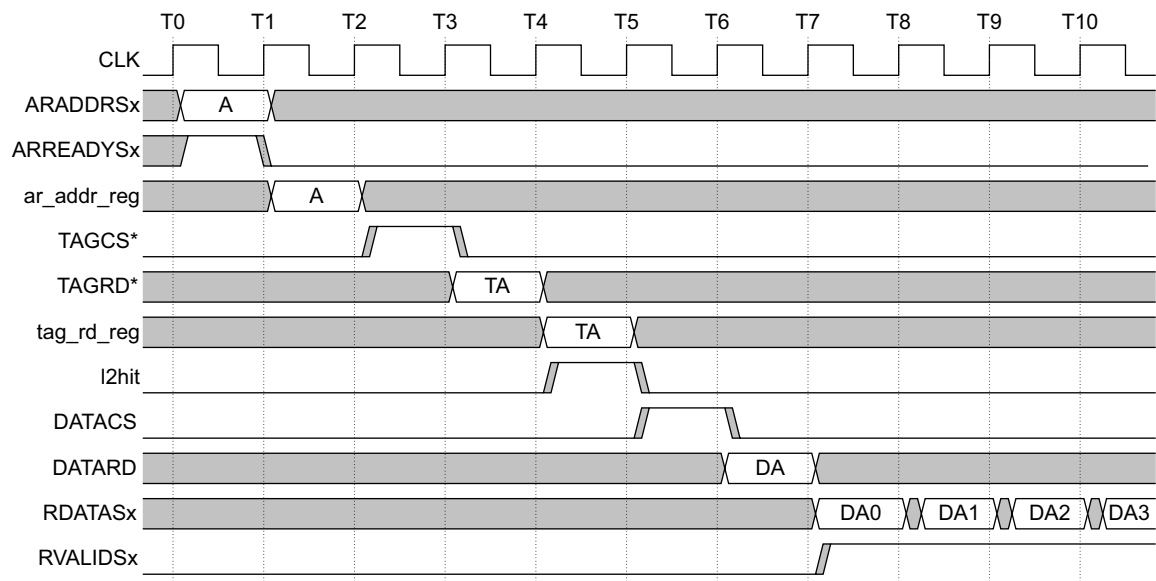


Figure C-1 Single read hit transaction

C.2 Single read miss transaction

Figure C-2 shows the timing for a single read miss transaction.

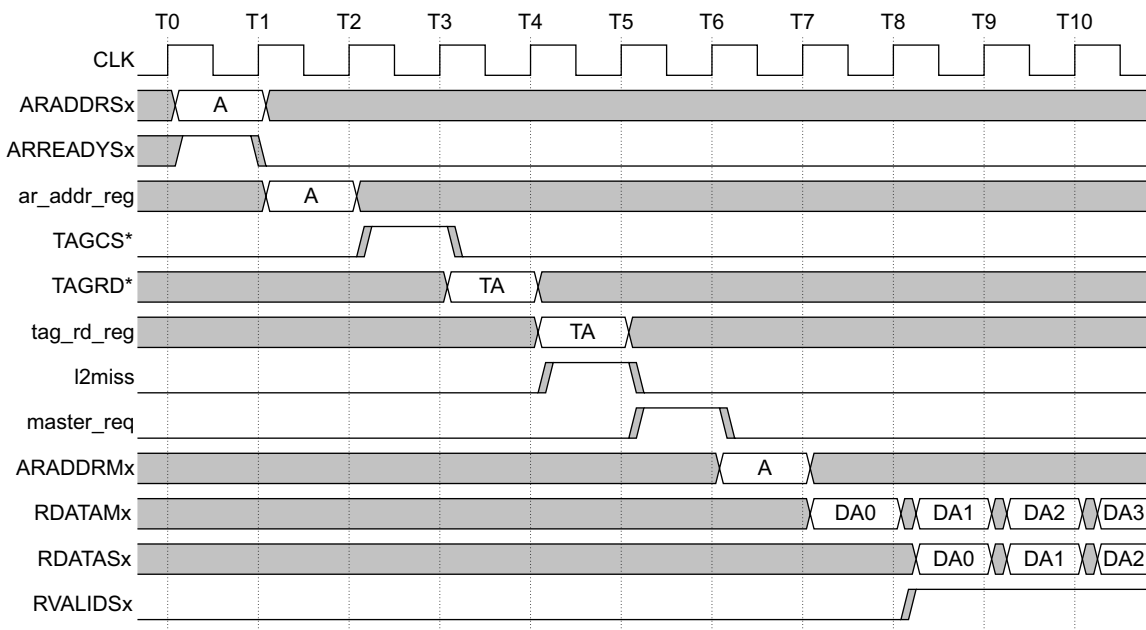


Figure C-2 Single read miss transaction

C.3 Single noncacheable read transaction

Figure C-3 shows the timing for a single noncacheable read transaction.

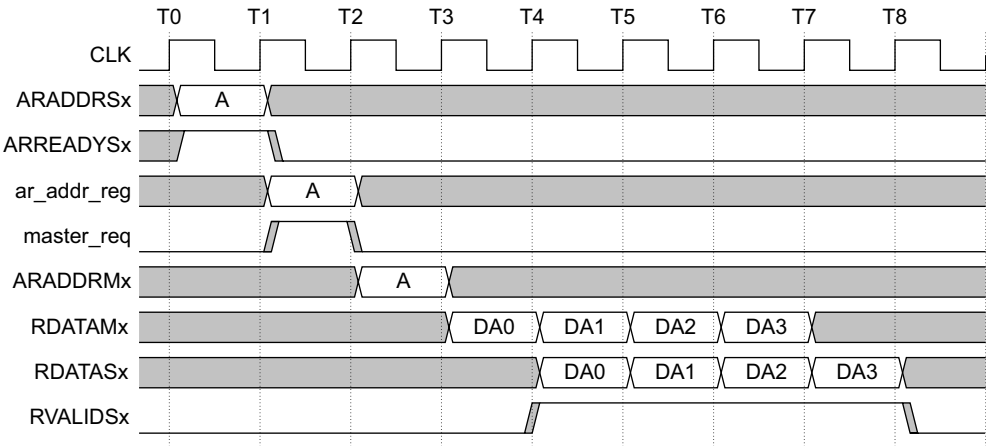


Figure C-3 Single noncacheable read transaction

C.4 Outstanding read hit transaction

Figure C-4 shows the timing for an outstanding read hit transaction.

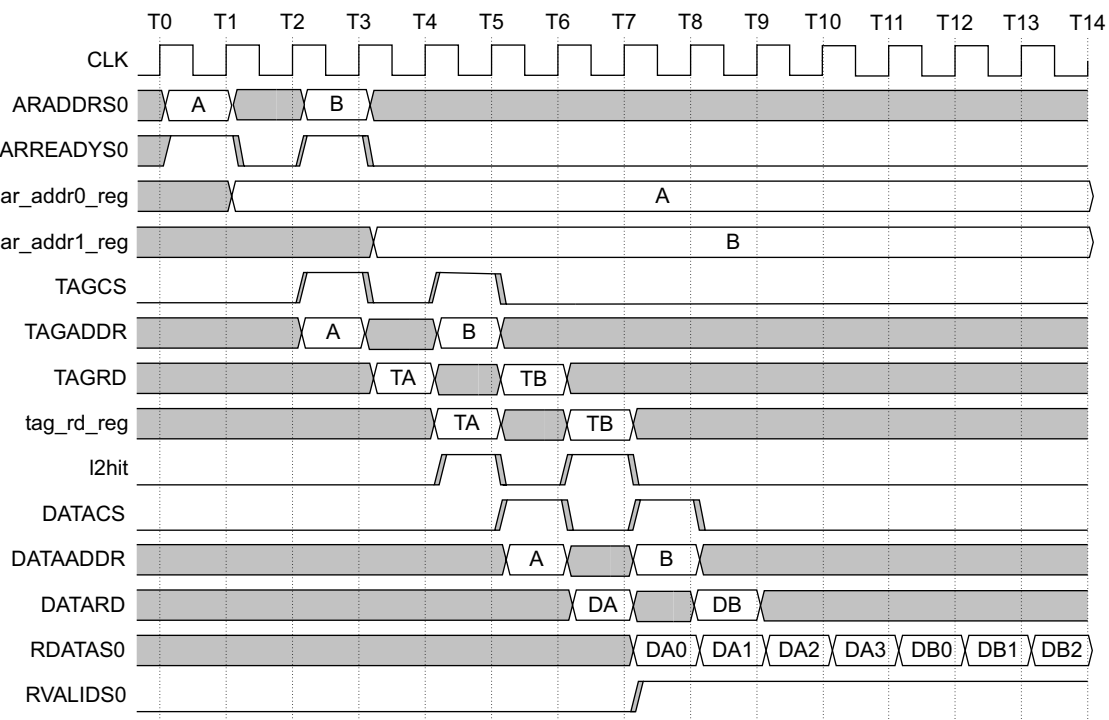


Figure C-4 Outstanding read hit transaction

C.5 Hit under miss read transactions

Figure C-5 shows the timing for a hit under miss read transaction case.

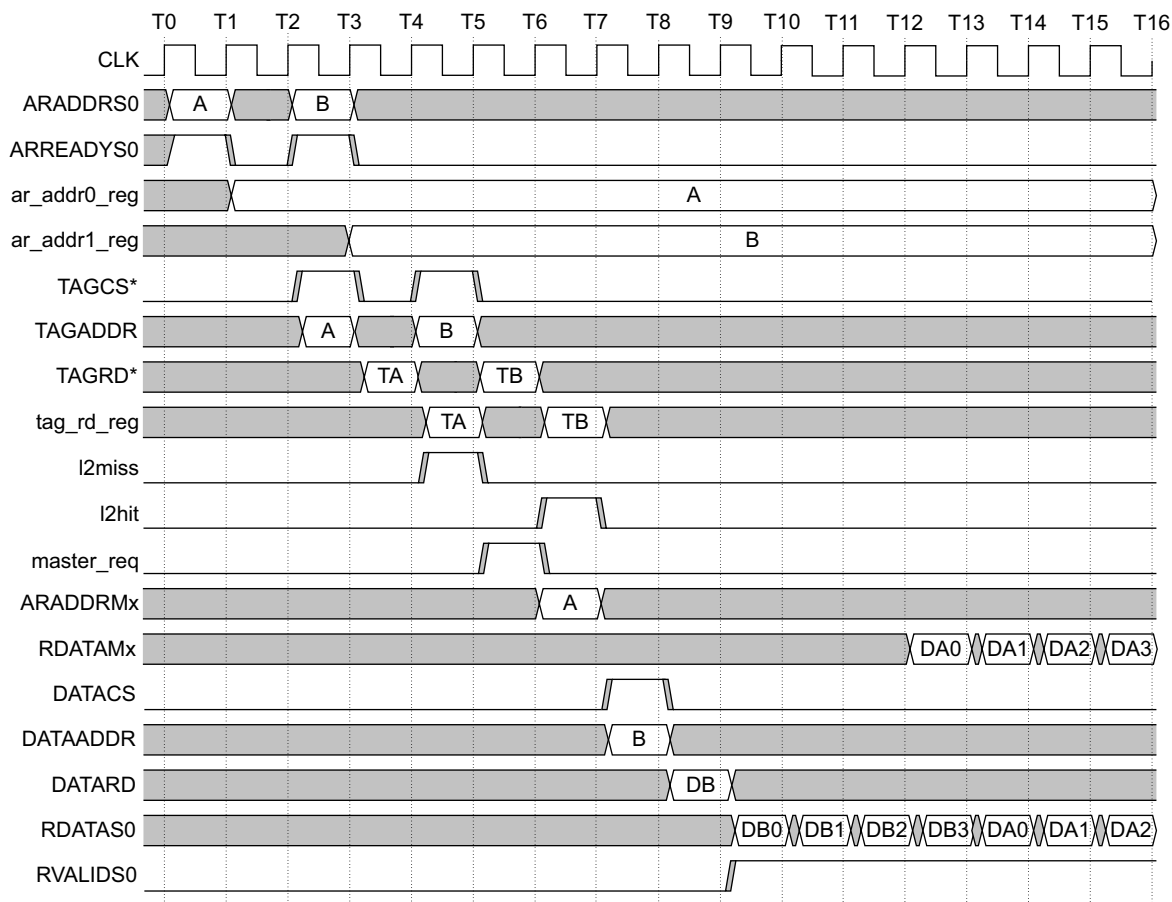


Figure C-5 Hit under miss read transaction

C.6 Single bufferable write transaction

Figure C-6 shows the timing for a single bufferable write transaction.

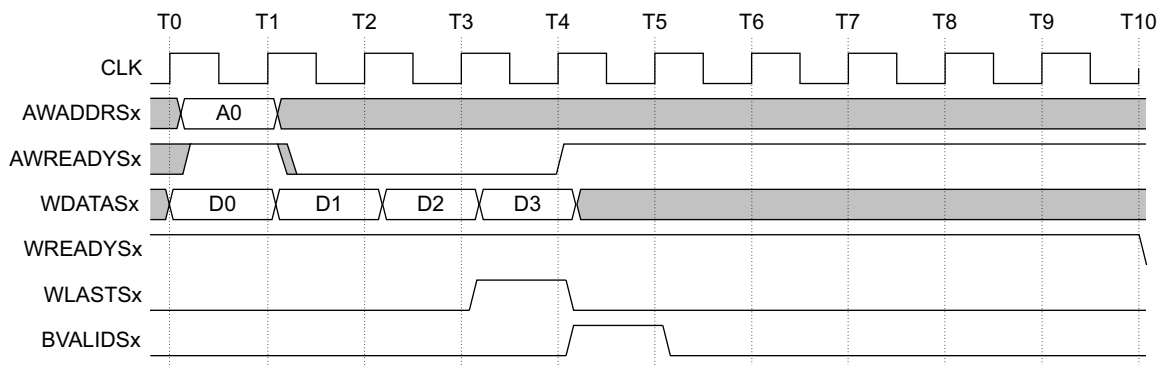


Figure C-6 Single bufferable write transaction

C.7 Single nonbufferable write transaction

Figure C-7 shows the timing for a single nonbufferable write transaction.

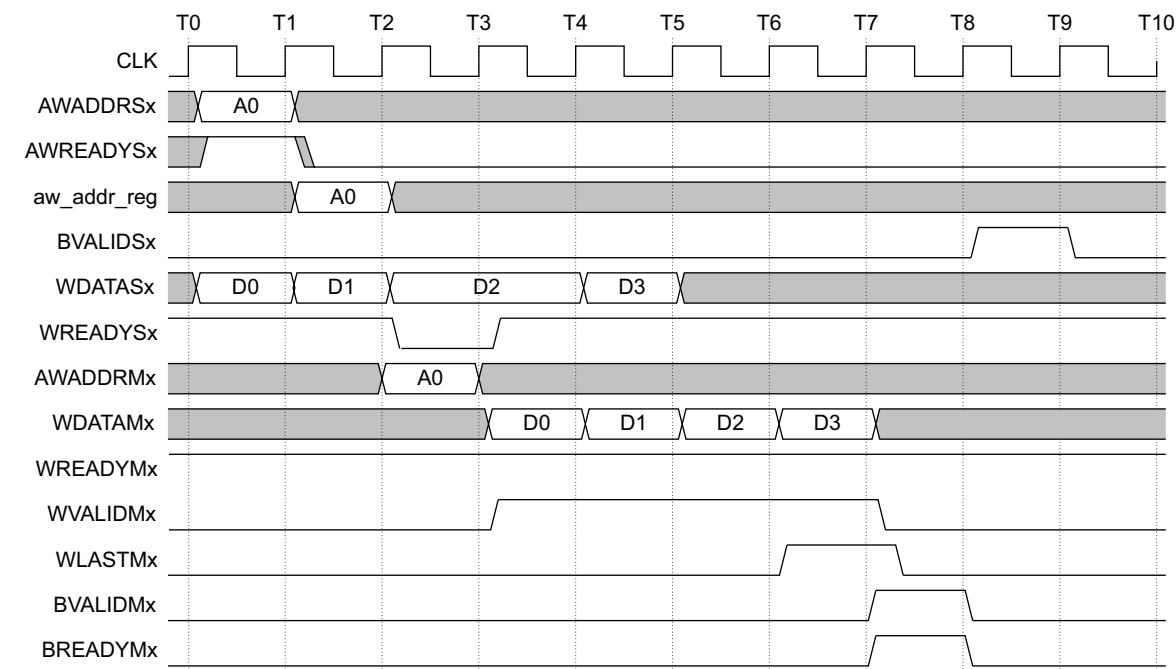


Figure C-7 Single nonbufferable write transaction

Appendix D

Revisions

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

Table D-1 Differences between issue B and issue C

Change	Location
Exported master ports AXI control signals added	Table 2-6 on page 2-9.
Master and slave port IDs updated	<i>Master and slave port IDs</i> on page 2-7.
Behavior of unmapped registers clarified	<i>Initialization sequence</i> on page 3-2.
Debug register access clarified	<ul style="list-style-type: none">• Table 3-1 on page 3-4• <i>Debug Register</i> on page 3-40.
System cache maintenance section added	<i>System Cache Maintenance Considerations</i> on page 3-30.
Lockdown by Master feature clarified	<i>Lockdown by master</i> on page 2-23.

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

Change	Location
TrustZone related behavior clarified	<i>TrustZone support in the cache controller on page 2-21.</i>
RAM clock and latencies clarified	<ul style="list-style-type: none">• Figure 2-9 on page 2-35• Figure 2-10 on page 2-35• Figure 2-13 on page 2-37.
Standby mode clarified	<i>Standby mode on page 2-50.</i>

Glossary

This glossary describes some of the terms used in this manual. Where terms can have several meanings, the meaning presented here is intended.

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 or Data Abort, and an internal or External Abort.

See also Data 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 Microcontroller Bus Architecture (AMBA)

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

AMBA

See Advanced Microcontroller Bus Architecture.

Application Specific Integrated Circuit (ASIC)

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.

ASIC

See Application Specific Integrated Circuit.

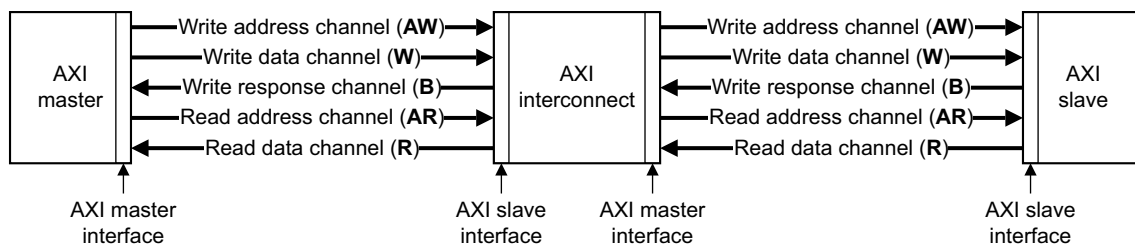
AXI

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.

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.

-
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.

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.

Beat

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

See also Burst.

Big-endian	<p>Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.</p> <p><i>See also</i> Little-endian and Endianness.</p>
Big-endian memory	<p>Memory in which:</p> <ul style="list-style-type: none"> • 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. <p><i>See also</i> Little-endian memory.</p>
Block address	<p>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.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Burst	<p>A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.</p> <p><i>See also</i> Beat.</p>
Byte	An 8-bit data item.
Byte lane strobe	A signal that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of this signal corresponds to eight bits of the data bus.
Cache	<p>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.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
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	<p>The basic unit of storage in a cache. It is always a power of two words in size (usually four or eight words), and is required to be aligned to a suitable memory boundary.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache line index	<p>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.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache lockdown	<p>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 are not subsequently reallocated. This ensures that all subsequent accesses to the instructions/data concerned are cache hits, and therefore complete as quickly as possible.</p>
Cache miss	<p>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.</p>
Cache set	<p>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. All sets are accessed in parallel during a cache look-up.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache way	<p>A group of cache lines (or blocks). It is 2 to the power of the number of index bits in size.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cast out	<p><i>See</i> Victim.</p>
Clean	<p>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.</p> <p><i>See also</i> Dirty.</p>
Coherency	<p><i>See</i> Memory coherency.</p>
Coprocessor	<p>A processor that supplements the main processor. It carries out additional functions that the main processor cannot perform. Usually used for floating-point math calculations, signal processing, or memory management.</p>
Copy back	<p><i>See</i> Write-back.</p>

Data Abort	<p>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.</p> <p><i>See also</i> Abort.</p>
Data cache	<p>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 reduce the average speed of memory accesses and so to increase processor performance.</p>
Dirty	<p>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.</p> <p><i>See also</i> Clean.</p>
Endianness	<p>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.</p> <p><i>See also</i> Little-endian and Big-endian</p>
Fully-associative cache	<p>A cache that has only one cache set that consists of the entire cache. The number of cache entries is the same as the number of cache ways.</p> <p><i>See also</i> Direct-mapped cache.</p>
Halfword	<p>A 16-bit data item.</p>
Index	<p><i>See</i> Cache index.</p>
Index register	<p>A register specified in some load or store instructions. The value of this register is used as an offset to be added to or subtracted from the base register value to form the virtual address, which is sent to memory. Some addressing modes optionally enable the index register value to be shifted prior to the addition or subtraction.</p>
Instruction cache	<p>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 greatly reduce the average time for memory accesses and so to increase processor performance.</p>
Invalidate	<p>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.</p>
Line	<p><i>See</i> Cache line.</p>

Little-endian	Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory. <i>See also</i> Big-endian and Endianness.
Little-endian memory	Memory in which: <ul style="list-style-type: none">• 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. <i>See also</i> 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.
Memory bank	One of two or more parallel divisions of interleaved memory, usually one word wide, that enable reads and writes of multiple words at a time, rather than single words. All memory banks are addressed simultaneously and a bank enable or chip select signal determines which of the banks is accessed for each transfer. Accesses to sequential word addresses cause accesses to sequential banks. This enables the delays associated with accessing a bank to occur during the access to its adjacent bank, speeding up memory transfers.
Memory coherency	A memory is coherent if the value read by a data read or instruction fetch is the value that was most recently written to that location. Memory coherency is made difficult when there are multiple possible physical locations that are involved, such as a system that has main memory, a store buffer, and a cache.
Microprocessor	<i>See</i> Processor.
Miss	<i>See</i> Cache miss.
Processor	A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

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.

See also Fast Context Switch Extension.

Read

Reads are defined as memory operations that have the semantics of a load. That is, the ARM instructions LDM, LDRD, LDC, LDR, LDRT, LDRSH, LDRH, LDRSB, LDRB, LDRBT, LDREX, RFE, STREX, SWP, and SWPB, and the Thumb instructions LDM, LDR, LDRSH, LDRH, LDRSB, LDRB, and POP. Java instructions that are accelerated by hardware can cause a number of reads to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.

Reserved

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

RAZ

Read As Zero.

SBO

See Should Be One.

SBZ

See Should Be Zero.

SBZP

See Should Be Zero or Preserved.

Set

See Cache set.

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.

Should Be Zero or Preserved (SBZP)

Should be written as 0 (or all 0s for bit fields) by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.

Store 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.

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.

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

Tightly coupled memory (TCM)

An area of low latency memory that provides predictable instruction execution or data load timing in cases where deterministic performance is required. TCMs are suited to holding:

- critical routines (such as for interrupt handling)
- scratchpad data
- data types whose locality is not suited to caching
- critical data structures (such as interrupt stacks).

TrustZone

This is a security extension for the ARM architecture.

Unaligned

Memory accesses that are not appropriately word-aligned or halfword-aligned.

See also Aligned.

UNP

See Unpredictable.

Unpredictable

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

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	Writes are defined as operations that have the semantics of a store. That is, the ARM instructions SRS, STM, STRD, STC, STRT, STRH, STRB, STRBT, STREX, SWP, and SWPB, and the Thumb instructions STM, STR, STRH, STRB, and PUSH. Java instructions that are accelerated by hardware can cause a number of writes to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.
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 completion	<p>The memory system indicates to the processor 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.</p> <p>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.</p>
Write-through (WT)	In a write-through cache, data is written to main memory at the same time as the cache is updated.
WT	<i>See</i> Write-through.

Cache terminology diagram

The diagram below illustrates the following cache terminology:

- block address
- cache line
- cache set
- cache way
- index
- tag.

