CoreLink™ Level 2 Cache Controller L2C-310

Revision: r3p3

Technical Reference Manual

ARM®

ARM DDI 0246H (ID080112)

CoreLink Level 2 Cache Controller L2C-310
Technical Reference Manual

Release Information

The following changes have been made to this book.

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
02 October 2009	D	Non-Confidential Unrestricted Access	First release for r3p0
03 February 2010	E	Non-Confidential Unrestricted Access	First release for r3p1
10 December 2010	F	Non-Confidential Unrestricted Access	First release for r3p2
13 July 2011	G	Non-Confidential Unrestricted Access	Second release for 3rp2
17 June 2012	H	Non-Confidential Unrestricted Access	First release for r3p3

Proprietary Notice

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Contents
CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual

Preface

This preface introduces the *CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual*. It contains the following sections:

- *About this book* on page vi
- *Feedback* on page ix.

About this book

This book is for the CoreLink Level 2 Cache Controller L2C-310.

Product revision status

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

r*n* Identifies the major revision of the product.

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

Intended audience

This book is written for hardware and software engineers implementing the CoreLink 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 book

This book 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

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

See *ARM Glossary*, `http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html`.

Conventions

Conventions that this book can use are described in:

- *Typographical conventions*
- *Timing diagrams*
- *Signals* on page viii.

Typographical conventions

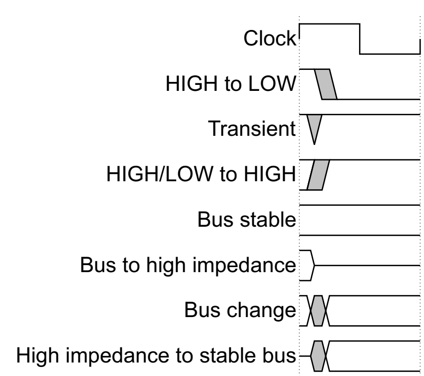
The following table describes the typographical conventions:

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace` *`italic`*	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

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

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



Key to timing diagram conventions

Signals

The signal conventions are:

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

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

Additional reading

This section lists publications by ARM and by third parties.

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

ARM publications

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

- *CoreLink Level 2 MBIST Controller L2C-310 Technical Reference Manual* (ARM DDI 0402)
- *CoreLink Level 2 Cache Controller L2C-310 Implementation Guide* (ARM DII 0045)
- *AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™* (ARM IHI 0022)
- *ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *Cortex™-A9 Technical Reference Manual* (ARM DDI 0388)
- *Cortex-A9 MPCore® Technical Reference Manual* (ARM DDI 0407).

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

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

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

Feedback on content

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

- the title
- the number, ARM DDI 0246H
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the CoreLink Level 2 Cache Controller L2C-310 and its features. It contains the following sections:

1.1 About the CoreLink Level 2 Cache Controller L2C-310

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 the processor generates significant memory traffic. By definition, a secondary cache assumes the presence of a Level 1 or primary cache, closely coupled to the processor or internal to the processor.

Memory access is fastest to the L1 cache, followed closely by the L2 cache. Memory access is typically significantly slower with the Level 3 (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	greater than 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 supports the following 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 1-way or 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 shows a top level diagram of the cache controller.

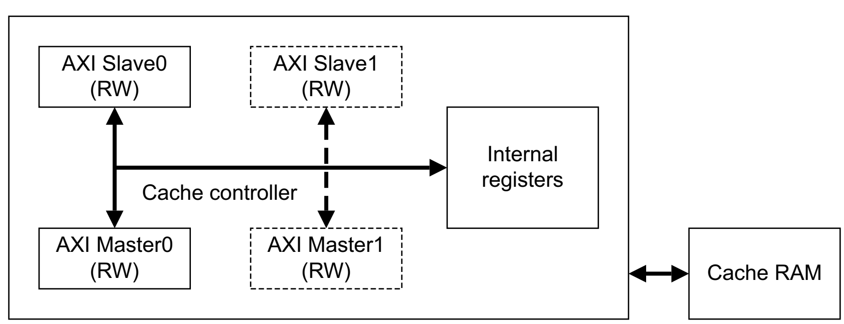


Figure 1-1 Top-level diagram

1.1.1 L2C-310 features

The L2C-310 cache controller supports the following features:

- Physical addressing and physical tagging.

- Lockdown format C, for data and instructions.

  Note

  Lockdown format C is also known as way locking.

- Lockdown by line.
- Lockdown by master ID.
- L2 cache available size can be 16KB to 8MB, depending on configuration and the use of the lockdown registers.
- Direct mapping 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.
- Banking on data RAM.
- 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.
- Treats normal memory non-cacheable shared reads as cacheable non-allocatable. Treats normal memory non-cacheable shared writes as cacheable write-through no write-allocate. The Shared Override option can override this behavior. See *Shareable attribute* on page 2-15.
- TrustZone 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 the security level of evictions to L3 memory system.
  - Restrictions for NS accesses for control, configuration, and maintenance registers to restrict access to secure data.
- Critical word first linefill.
- Pseudo-Random, or round-robin 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 and wait 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 for a 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.

- 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 2-11.
- Prefetching capability. See *Auxiliary Control Register* on page 3-10.
- Integer, 1:1, 2:1 clock ratios and half-integer, 1.5:1, 2.5:1, and 3.5:1 clock ratios controlled by clock enable inputs on slave and master ports.
- Wait, latency, clock enable, parity, and error support at the RAM interfaces.
- *Memory Built In Self Test* (MBIST).
- L2 cache event monitoring. Exports event signals if you require to use them in conjunction with an event monitoring block. Event monitoring is also available 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 master port.

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

Table 1-2 RTL options

Feature	RTL option
16-way associativity	`pl310_16_WAYS`
Data RAM banking	`pl310_DATA_BANKING`
Number of slave ports	`pl310_S1`
Number of master ports	`pl310_M1`, requires `pl310_S1`
Parity	`pl310_PARITY`
Address filtering	`pl310_ADDRESS_FILTERING`, requires `pl310_M1`
Lockdown by master	`pl310_LOCKDOWN_BY_MASTER`
Lockdown by line	`pl310_LOCKDOWN_BY_LINE`
Speculative read logic	`pl310_SPECULATIVE_READ`
Slave AXI ID width	`pl310_AXI_ID_MAX`

RAM latencies	`pl310_TAG_SETUP_LAT`
	`pl310_TAG_READ_LAT`
	`pl310_TAG_WRITE_LAT`
	`pl310_DATA_SETUP_LAT`
	`pl310_DATA_READ_LAT`
	`pl310_DATA_WRITE_LAT`

Presence of **ARUSERMx** and **AWUSERMx** sideband signals	`pl310_ID_ON_MASTER_IF`

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.

You configure the cache controller using memory-mapped registers, rather than using CP15 instructions. See Chapter 3 *Programmers Model*.

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

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

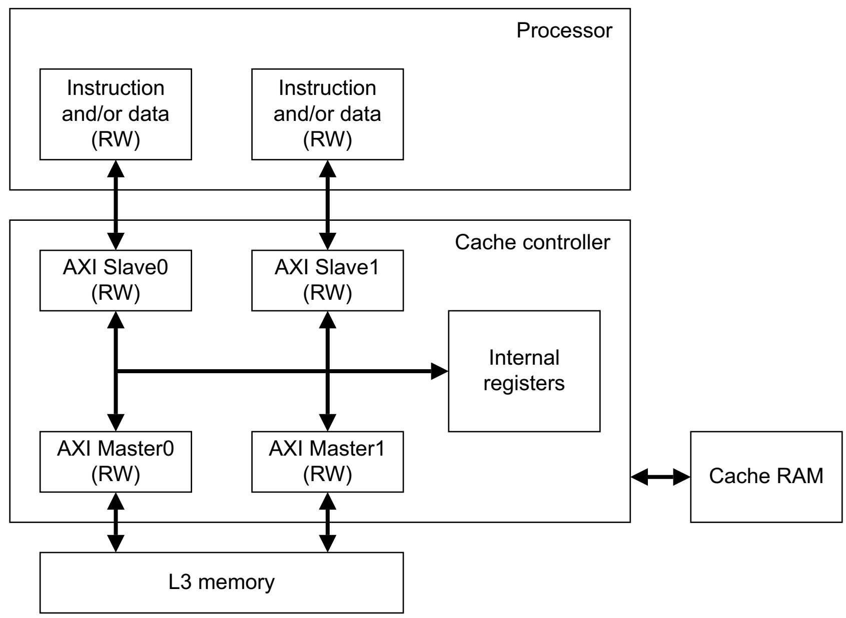


Figure 1-2 Example cache controller interfaced to an ARM processor

You can configure the cache controller to use one or two master ports. Table 1-3 shows what the L2C-310 uses each master port 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** or **S1**	Linefills from **S0** or **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 you implement and enable address filtering:
  - 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** and **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.

r2p0-r3p0 The main differences between these versions are:

- new data RAM banking feature
- new speculative read optimization with Cortex-A9 MPCore processor
- new low-power modes
- new AXI ID encodings on master interfaces
- support for 64-byte linefills issued to L3 memory system.

r3p0-r3p1 The main differences between these versions are:

- errata fixes, see the errata notice for more information
- new sideband signals on master interfaces, and new synthesis option to control their implementation.

r3p1-r3p2 There is no functional difference between these two revisions.

r3p2-r3p3 There is no functional difference between these two revisions.

Chapter 2
Functional Overview

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

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 **WAYSIZE[2:0]** input	16KB, 32KB, 64KB, 128KB, 256KB, 512KB	`0b001`	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	`0b111`	8 cycles of latency
Data RAM banking	Verilog `` `define pl310_DATA_BANKING ``	Commented or uncommented	Commented	No data RAM banking
Slave port 1 present	Verilog `` `define pl310_S1 ``	Commented or uncommented	Commented	No slave port 1
Master port 1 present	Verilog `` `define pl310_M1 ``	Commented or uncommented	Commented	No master port 1
Parity logic	Verilog `` `define pl310_PARITY ``	Commented or uncommented	Commented	No parity logic
Lockdown by master	Verilog `` `define pl310_LOCKDOWN_BY_MASTER ``	Commented or uncommented	Commented	No lockdown by master
Lockdown by line	Verilog `` `define pl310_LOCKDOWN_BY_LINE ``	Commented or uncommented	Commented	No lockdown by line
AXI ID width on slave ports	Verilog `` `define pl310_AXI_ID_MAX <value> ``	>=2	5	6 AXI ID bits on slave ports and 8 on master ports
Address filtering	Verilog `` `define pl310_ADDRESS_FILTERING ``	Commented or uncommented	Commented	No address filtering logic
Speculative read	Verilog `` `define pl310_SPECULATIVE_READ ``	Commented or uncommented	Commented	No logic for supporting speculative read
Presence of **ARUSERMx** and **AWUSERMx** sideband signals	Verilog `` `define pl310_ID_ON_MASTER_IF ``	Commented or uncommented	Commented	No sideband signals

a. 16-way associativity must be enabled using the `pl310_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 2-5
- *Master and slave port IDs* on page 2-7
- *Exported AXI control* on page 2-8
- *AXI locked and exclusive accesses* on page 2-9.

Note

The cache controller reaches optimal performance when it receives AXI transactions that target full cache lines. That is, when all of the following conditions are met:

- **AxLENSy** = `0x3`
- **AxSIZESy** = `0x3`
- one of the following conditions is true:
  - **AxBURSTSy** = `0x1` and **AxADDRSy[4:3]** = `0x0`
  - **AxBURSTSy** = `0x2`.

Where x = R or Y, and y = 0 or 1.

You can obtain optimal performance by using different AXI IDs for outstanding read transactions sent to the L2C-310 slave interfaces.

2.2.1 AXI master and slave interface attributes

Table 2-2 shows the AXI master interface attributes. The attribute values are maximum values. They might not be reached in all systems.

Table 2-2 AXI master interface attributes

Configuration	Attribute	Value
Two master ports	Write issuing capability	12, consisting of: • 6 evictions • 6 writes from store buffer
	Read issuing capability	11, consisting of: • 8, 4 per master port, prefetches or reads from slave ports • 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: • 6 evictions • 6 writes from store buffer
	Read issuing capability	7, consisting of: • 4 prefetches or reads from one or more slave ports • 3 reads from store buffer
	Combined issuing capability	19
	Write interleave capability	1
	Write ID width	Defined by Write ID width on slave ports
	Read ID width	Defined by Read ID width on slave ports

Table 2-3 shows the AXI slave interface attributes. The attribute values in Table 2-3 are maximum values. They might not be reached in all systems.

Table 2-3 AXI slave interface attributes

Configuration	Attribute	Value
Two slave ports and two master ports[a]	Write acceptance capability	6, 3 per slave port
	Read acceptance capability	• 16, 8 per slave port • 24, 12 per slave port, for sequential accesses, when you enable the double linefill feature
	Combined acceptance capability	• 22 • enable the double linefill feature to increase this maximum value
	Write interleave depth	2, 1 per slave port
	Read data reorder depth	16
	Write ID width	Parameterizable, defined by `pl310_AXI_ID_MAX`, default is 6
	Read ID width	Parameterizable, defined by `pl310_AXI_ID_MAX`, default is 6
Two slave ports and one master port	Write acceptance capability	6, 3 per slave port
	Read acceptance capability	• 12, 6 per slave port • 16, 8 per slave port, for sequential accesses, when you enable the double linefill feature
	Combined acceptance capability	• 18 • 22 when you enable the double linefill feature.
	Write interleave depth	2, 1 per slave port
	Read data reorder depth	12
	Write ID width	Parameterizable, defined by `pl310_AXI_ID_MAX`, default is 6
	Read ID width	Parameterizable, defined by `pl310_AXI_ID_MAX`, default is 6

Table 2-3 AXI slave interface attributes (continued)

Configuration	Attribute	Value
One slave port and one master port	Write acceptance capability	3
	Read acceptance capability	• 8 • enable the double linefill feature to increase this value to 12 for sequential accesses
	Combined acceptance capability	• 11 • 15 when you enable the double linefill feature
	Write interleave depth	1
	Read data reorder depth	8
	Write ID width	Parameterizable, defined by `pl310_AXI_ID_MAX`, default is 6
	Read ID width	Parameterizable, defined by `pl310_AXI_ID_MAX`, default is 6

a. These values ignore address filtering. If you implement address filtering, the values might be reduced to values close to those in the *Two slave ports and one master port* section of the table.

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 from **CLK**. The ratios 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 drive both cache controller inputs, **INCLKEN** and **OUTCLKEN**. See Figure 2-1.

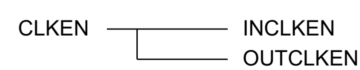


Figure 2-1 CLKEN used to drive cache controller inputs in case of integer clock ratio

Figure 2-2 on page 2-6 shows how the different clock enable signals can be generated to support half integer clock ratios, where **ACLK** is the clock of the L3 AXI system.

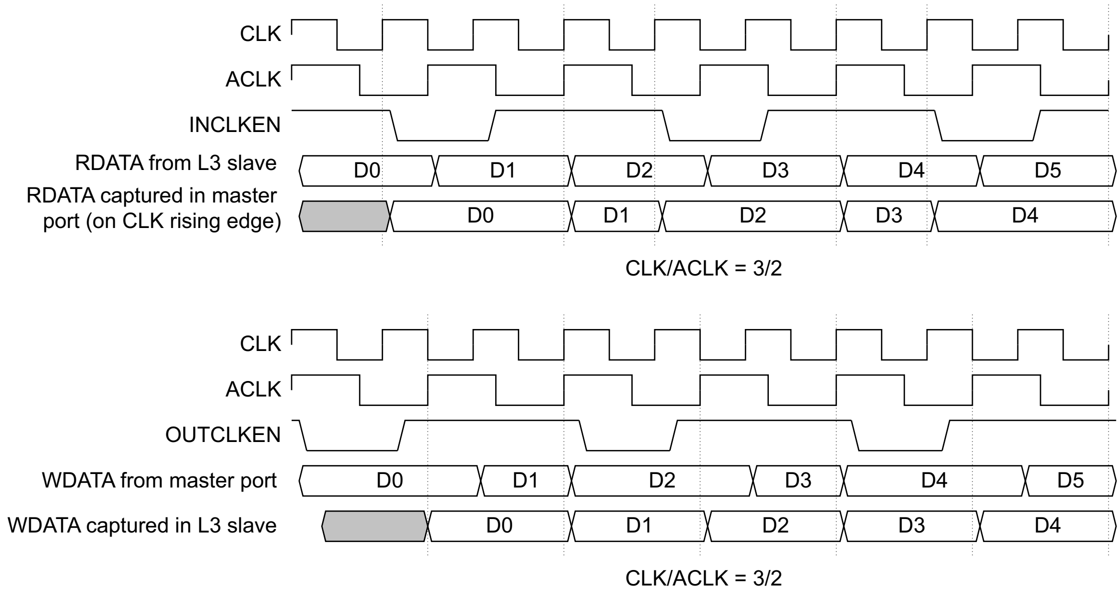


Figure 2-2 Clock enable usage model for 1.5:1 clock ratio in master port

Figure 2-3 shows how the different clock enable signals can be generated to support half integer clock ratios. In Figure 2-3, **ACLK** is the clock of the L3 AXI system.

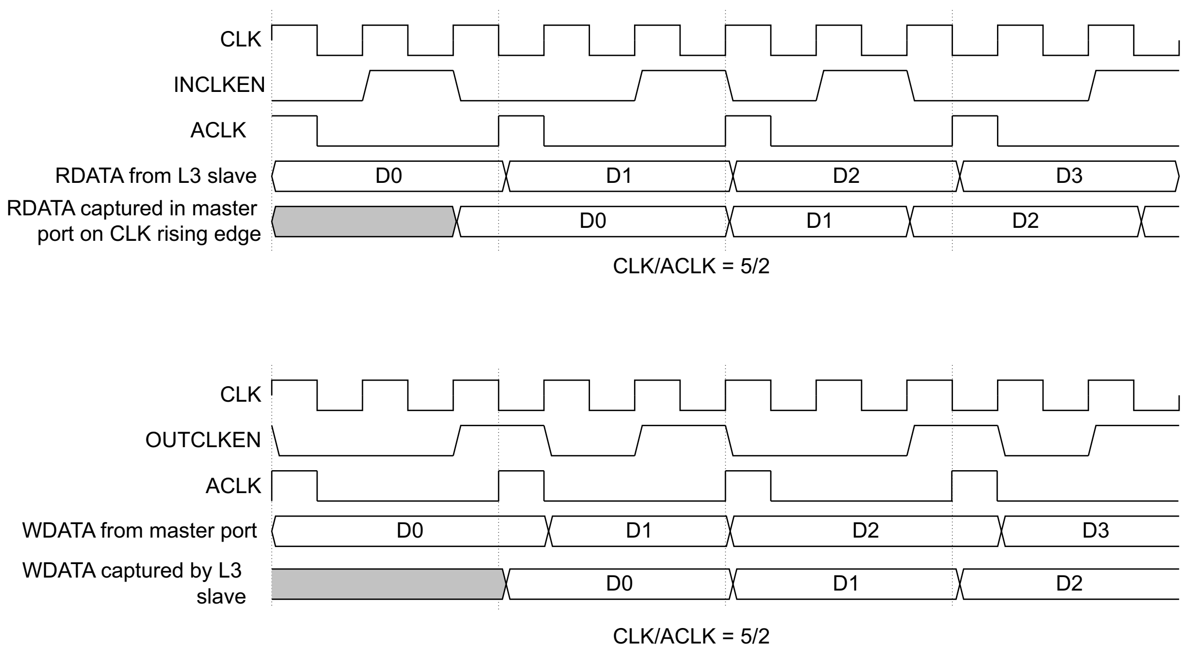


Figure 2-3 Clock enable usage model for 2.5:1 clock ratio in master port

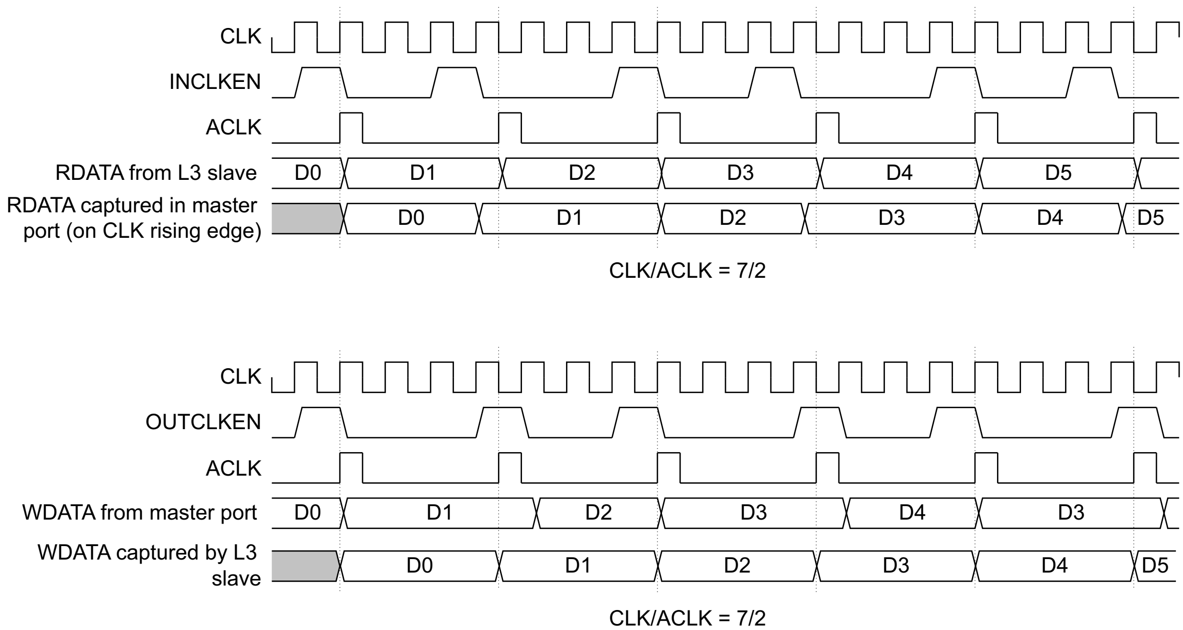


Figure 2-4 Clock enable usage model for 3.5:1 clock ratio in master port

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

In each of these AXI interfaces, the controller uses the clock enable as follows:

- samples inputs on the rising edge of **CLK** only when **INCLKEN** is HIGH.
- updates outputs on the rising edge 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 `pl310_AXI_ID_MAX` that has a default value of 5. The AXI ID width is `` [`pl310_AXI_ID_MAX:0] `` on the slave ports and `` [`pl310_AXI_ID_MAX+2:0] `` on the master ports. This section assumes that the default value for `pl310_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**.

2.2.4 Exported AXI control

Table 2-4 shows information on the AXI control signals on the master ports of the cache controller.

Table 2-4 Exported master ports AXI control signals

Access type	Master control signals buses	Value, Verilog
Non-cacheable read transactions, non-bufferable write transactions, or cache disabled	**AWCACHEMx** and **ARCACHEMx** **AWPROTMx** and **ARPROTMx** **AWLOCKMx** and **ARLOCKMx** **AWADRRMx** and **ARADDRMx** **AWLENMx** and **ARLENMx** **AWSIZEMx** and **ARSIZEMx** **AWBURSTMx** and **ARBURSTMx**	All as original transaction
Linefills associated with a cacheable read transaction	**ARCACHEMx**	As original transaction
	ARPROTMx	As original transaction
	ARLOCKMx	{00}
	ARADDRMx[2:0]	`0b000`
	ARLENMx	`0b0011` or `0b0111` if double linefill is enabled
	ARSIZEMx	`0b11`, 64-bit
	ARBURSTMx	`0b10`, WRAP, or `0b01`, INCR
Read or write from store buffer, including write-through	**ARCACHEMx** and **AWCACHEMx**	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
	ARPROTMx and **AWPROTMx**	{0, SECURITY[a], 1}
	ARLOCKMx and **AWLOCKMx**	{00}
	ARADDRMx[2:0]	`0b000`
	ARLENMx	`0b0011`
	ARSIZEMx	`0b11`, 64-bit
	ARBURSTMx	`0b01`, INCR
	AWADDRMx[2:0]	• `0b000` if Normal Memory • depends on original transaction if Device Memory.
	AWLENMx	between `0b0000` and `0b0011`
	AWSIZEMx	• `0b11`, 64-bit if Normal Memory • depends on original transaction if Device Memory.
	AWBURSTMx	`0b01`, INCR

Table 2-4 Exported master ports AXI control signals (continued)

Access type	Master control signals buses	Value, Verilog
Evictions	**AWCACHEMx**	{1111}
	AWPROTMx	{0, SECURITY[a], 1}
	AWLOCKMx	{00}
	AWADDRMx[3:0]	`0b0000`
	AWLENMx	`0b0011`
	AWSIZEMx	`0b11`, 64-bit
	AWBURSTMx	`0b01`, INCR

a. Denotes the value of the NS attribute stored with the data.

——— Note ———

In some circumstances, the L2 cache controller can issue writes to the L3 memory system with sparse strobes, or even with WSTRBMx = `0b00000000`.

2.2.5 AXI locked and exclusive accesses

The following sections describe AXI locked and exclusive accesses:

- *AXI locked transfers*
- *AXI exclusive accesses* on page 2-10.

AXI locked transfers

For a non-cacheable transfer, the access is forwarded to L3 memory through the master ports and is marked as locked.

For cacheable transfers, the controller always performs a cache lookup. For a cache miss, it requests a linefill, non-locked, on the master side. Write accesses always cause non-locked writes on the master side.

When a slave performs a locked sequence, cacheable or non-cacheable, the controller stops the other slave from accepting more transfers. It stalls a locked transaction until all buffers, including the store buffer, are empty.

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 arrive at the same time, the controller permits them 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 either non-cacheable or cacheable transactions. It cannot contain a mix of cacheable and non-cacheable 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 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 cache controller can return an EXOKAY response:

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

  Note

  In this context the word cacheable refers to whether a lookup to the L2 cache is done, so this might include shareable non-cacheable accesses. See *Shareable attribute* on page 2-15.

- 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, for example 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.

The AXI specification requires that control signals in the read and write parts of an exclusive sequence must be identical. This includes the AXI ID. However, the L2C-310 enables you to issue different AXI IDs between the read and write parts of a non-cacheable exclusive sequence when the accesses are sent to the L3 memory system. If the exclusive monitor located in the L3 memory system supports this behavior, this enables you to maximize the performance during the exclusive sequence.

Note

This behavior is not fully compatible with AXI. You control it using bit 21 of the Prefetch Control Register. See *Prefetch Control Register* on page 3-34.

2.3 Cache operation

Table 2-5 to Table 2-13 on page 2-13 show the general behavior of the cache controller for different ARMv6 and ARMv7 transactions.

Table 2-5 shows the general behavior of the cache controller for non-cacheable and non-bufferable AXI transactions.

Table 2-5 Non-cacheable and non-bufferable 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 2-6 shows the general behavior of the cache controller for bufferable only AXI transactions.

Table 2-6 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 L2C-310 treats a Device write that crosses a cache line boundary as a Strongly Ordered access.

Table 2-7 shows the general behavior of the cache controller for cacheable but do not allocate AXI transactions, and cacheable and bufferable but do not allocate AXI transactions.

Table 2-7 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 2-8 shows the general behavior of the cache controller for cacheable write-through, allocate on read AXI transactions.

Table 2-8 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 2-9 shows the general behavior of the cache controller for cacheable write-back, allocate on read AXI transactions.

Table 2-9 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 2-10 shows the general behavior of the cache controller for cacheable write-through, allocate on write AXI transactions.

Table 2-10 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: • 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 2-11 shows the general behavior of the cache controller for cacheable write-back, allocate on write AXI transactions.

Table 2-11 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: • 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 2-12 shows the general behavior of the cache controller for cacheable write-through, allocate on read and write AXI transactions.

Table 2-12 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: • 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 read and write AXI transactions.

Table 2-13 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: • 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 2-5 on page 2-11 to Table 2-13 using parameters such as shareable attribute, Force write allocate, and exclusive cache configuration.

Other behaviors are described in:

- *Shareable attribute* on page 2-15
- *Force write allocate* on page 2-16
- *Exclusive cache configuration* on page 2-17.

2.3.1 Cache attributes

Table 2-14 shows the **AWCACHE[3:0]** and **ARCACHE[3:0]** signals as the *AMBA AXI Protocol Specification* defines, and the ARMv6 and ARMv7 equivalent meaning. Table 2-14 does not show AXI locked and exclusive accesses.

Table 2-14 AWCACHE and ARCACHE definitions

AWCACHE and ARCACHE				AXI meaning	ARMv6 and ARMv7 equivalent
WA	**RA**	**C**	**B**		
0	0	0	0	Non-cacheable, non-bufferable	Strongly ordered
0	0	0	1	Bufferable only	Device
0	0	1	0	Cacheable but do not allocate	Outer non-cacheable
0	0	1	1	Cacheable and bufferable, do not allocate	Outer non-cacheable
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

- This table does not describe the shareable attribute **AyUSERSx[0]**, where y = R or W, and x = 0 or 1. *Shareable attribute* on page 2-15 describes its behavior.
- 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 Shareable attribute

The **ARUSERSx[0]** and **AWUSERSx[0]** signals affect transactions. Typically, ARM processors drive these signals and reflect the shareable attribute as defined in the ARMv6 and ARMv7 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 cache controller ignores the shareable attribute.

The default behavior of the cache controller with respect to the shareable 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 the L3 memory system. 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 that support 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 Shareable attribute override enable bit without flushing the cache could cause a hazard where incorrect data could be evicted causing more recent data in the L3 memory system to be overwritten.
- The behavior of the L2C-310 with respect to the shareable attribute is different from the L220 Level 2 cache controller. Take care when moving from an L220 based system to a system implementing L2C-310 because the shareable 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[24:23] in the Auxiliary Control Register is 00. This configures the cache controller to use the received **AWCACHE** or **ARCACHE** attributes. You can find additional reference data on the general behavior in a set of tables. See Table 2-5 on page 2-11 to Table 2-14 on page 2-14.

If you set the Force write allocate bits in the Auxiliary Control Register to `0b01`, the cache controller never allocates cacheable write misses into the cache.

If you set the Force write allocate bits in the Auxiliary Control Register to `0b10`, the cache controller always allocates cacheable write misses into 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 feature has priority over the exclusive cache configuration behavior described in *Exclusive cache configuration* on page 2-17.

2.3.4 Exclusive cache configuration

Note

You must only enable this configuration if:

- the processor that drives the cache controller also supports this feature
- you also enable this feature in the processor.

Setting the exclusive cache configuration bit[12] 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, **AyCACHESx** = **AyUSER[4:1]** = `0b1011` or `0b0111` or `0b1111` where y = R or W, and x = 0 or 1.

Reads

For reads, the behavior is as follows:

- For a hit, the cache controller marks the line as non-valid, that is, it resets the tag RAM valid bit, 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.
- For a miss, the cache controller does not allocate the line 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.

- For a hit, the line is marked dirty unless **AWUSERSx[9:8]** = `0b11`. In this case, the dirty bit is unchanged.
- For 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 memory system 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 L2C-310:

- 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 `pl310_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 `pl310_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 3-28. 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 3-28.

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 3-29 and Table 3-21 on page 3-29.

Lockdown by master

The Lockdown by master feature is a superset of the Lockdown by way feature. It enables multiple masters to share the L2 cache and makes the L2 cache behave as though these masters have dedicated smaller L2 caches.

This feature enables you to reserve ways of the L2 cache to specific L1 masters that the ID on the **AyUSERSx[7:5]** signals identify, where y = R or W, and x = 0 or 1.

You can use sixteen registers for controlling this mechanism. See the tables from Table 3-20 on page 3-29 to Table 3-35 on page 3-31.

System including four Cortex-A9 MPCore processors and L2C-310 and *System including one Cortex-A9 MPCore processor with ACP and L2C-310* on page 2-20 describe two examples of a system that use the lockdown by master. They show you how to drive **AyUSERSx[7:5]** signals and program the relevant registers.

Note

The settings for the Lockdown registers in *System including four Cortex-A9 MPCore processors and L2C-310* and *System including one Cortex-A9 MPCore processor with ACP and L2C-310* on page 2-20 are provided to ensure the best performance when considering the cache replacement policy implemented in the L2C-310.

System including four Cortex-A9 MPCore processors and L2C-310

Figure 2-5 shows a system where a Cortex-A9 cluster with four CPUs drives the L2C-310.

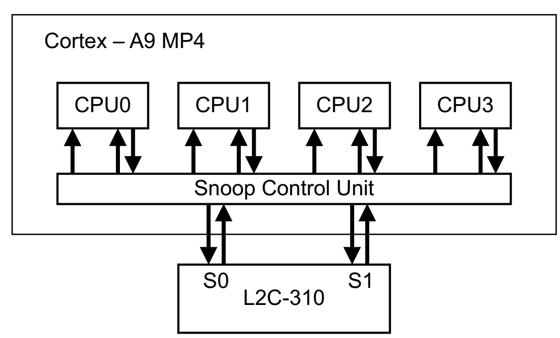


Figure 2-5 Driven by cortex a9 cluster with 4 CPUs

If you implement the L2C-310 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 **AyUSERSx[7:5]** signals as follows:

AyUSERSx[7:5] = {0b0, **AyIDMx[1:0]**}.

Note

AyIDMx[1:0] are the AXI ID bits that the Cortex-A9 cluster drives. See the *Cortex-A9 Technical Reference Manual*.

Table 2-15 shows how you can program the lockdown registers.

Table 2-15 MP4 system lockdown register definitions

Register	Offset	Value
Data Lockdown 0[a]	`0x900`	`0x0000EEEE`
Instruction Lockdown 0	`0x904`	`0x0000EEEE`
Data Lockdown 1	`0x908`	`0x0000DDDD`
Instruction Lockdown 1	`0x90C`	`0x0000DDDD`
Data Lockdown 2	`0x910`	`0x0000BBBB`
Instruction Lockdown 2	`0x914`	`0x0000BBBB`
Data Lockdown 3	`0x918`	`0x00007777`
Instruction Lockdown 3[b]	`0x91C`	`0x00007777`
All other Lockdown registers	`ox920-0x93C`	default

a. Used when **AyUSERSx[7:5]** = 000, transactions from CPU0, and **AyPROTSx[2]** = 0.

b. Used when **AyUSERSx[7:5]** = 011, transactions from CPU3, and **AyPROTSx[2]** = 1.

This configuration reserves:

- ways [0], [4], [8], and [12] for CPU0
- ways [1], [5], [9], and [13] for CPU1
- ways [2], [6], [10], and [14] for CPU2
- ways [3], [7], [11], and [15] for CPU3.

System including one Cortex-A9 MPCore processor with ACP and L2C-310

Figure 2-6 shows a system where the Cortex-A9 cluster has only one CPU and the *Accelerator Coherence Port* (ACP) drives the L2C-310.

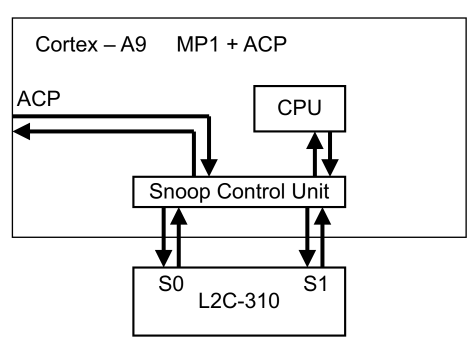


Figure 2-6 Driven by cortex a9 cluster with 1 CPU and ACP

If you implement the L2C-310 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 **AyUSERSx[7:5]** signals as follows:

- **AyUSERSx[7:5]** = {0b00, **AyIDMx[2]**}.

Note

AyIDMx are the AXI ID bits that the Cortex-A9 cluster drives. See the *Cortex-A9 Technical Reference Manual*.

Table 2-16 shows how you can program the lockdown registers.

Table 2-16 MP1 plus APC system lockdown register definitions.

Register	Offset	Value
Data Lockdown 0[a]	`0x900`	`0x000000AA`
Instruction Lockdown 0	`0x904`	`0x000000AA`
Data Lockdown 1	`0x908`	`0x00000055`
Instruction Lockdown 1[b]	`0x90C`	`0x00000055`
All other Lockdown registers	`ox910 - 0x93C`	default

a. Used when **AyUSERSx[7:5]** = 000, transactions from CPU, and **AyPROTSx[2]** = 0.

b. Used when **AyUSERSx[7:5]** = 001, transactions from ACP, and **AyPROTSx[2]** = 1.

When you have programmed the registers, and the cache controller drives **AyUSERSx[7:5]** as *Lockdown by way* on page 2-18 shows, the cache controller reserves:

- ways [0], [2], [4], and [6] for the Cortex-A9 CPU
- ways [1], [3], [5], and [7] for the ACP.

2.4 RAM interfaces

This section describes:
- *RAM organization*
- *RAM clocking and latencies* on page 2-30
- *MBIST support* on page 2-33.

2.4.1 RAM organization

This section describes:
- *Advantages of data RAM banking*
- *Data RAM without banking* on page 2-23
- *Data RAM with banking* on page 2-24
- *Data parity RAM without banking* on page 2-25
- *Data parity RAM with banking* on page 2-26
- *Tag RAM* on page 2-26
- *Tag parity RAM* on page 2-27
- *RAM bus usage versus cache associativity and way size* on page 2-27.

Advantages of data RAM banking

Timing closure is difficult with designs that have a high clock frequency and large L2 cache, especially for the data RAM. Such systems require high RAM latencies, that reduce the performance of the system. To counter this effect, you can split the data RAM into four banks. This feature enables pipelined accesses to the data RAM.

Figure 2-7 and Figure 2-8 show the benefit of the banking when two consecutive reads targeting different banks are treated with the following programmed latencies:
- Data RAM setup latency = 2 cycles, programmed value = `0x1`
- Data RAM read latency = 4 cycles, programmed value = `0x3`.

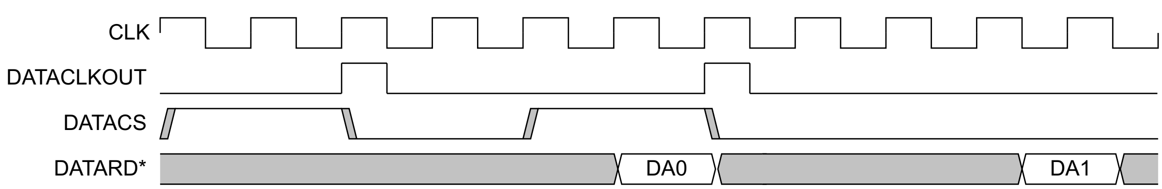


Figure 2-7 No multi-banking

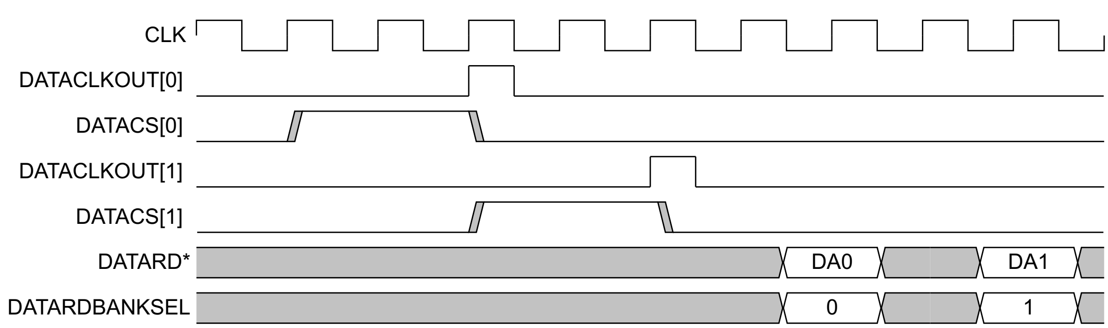
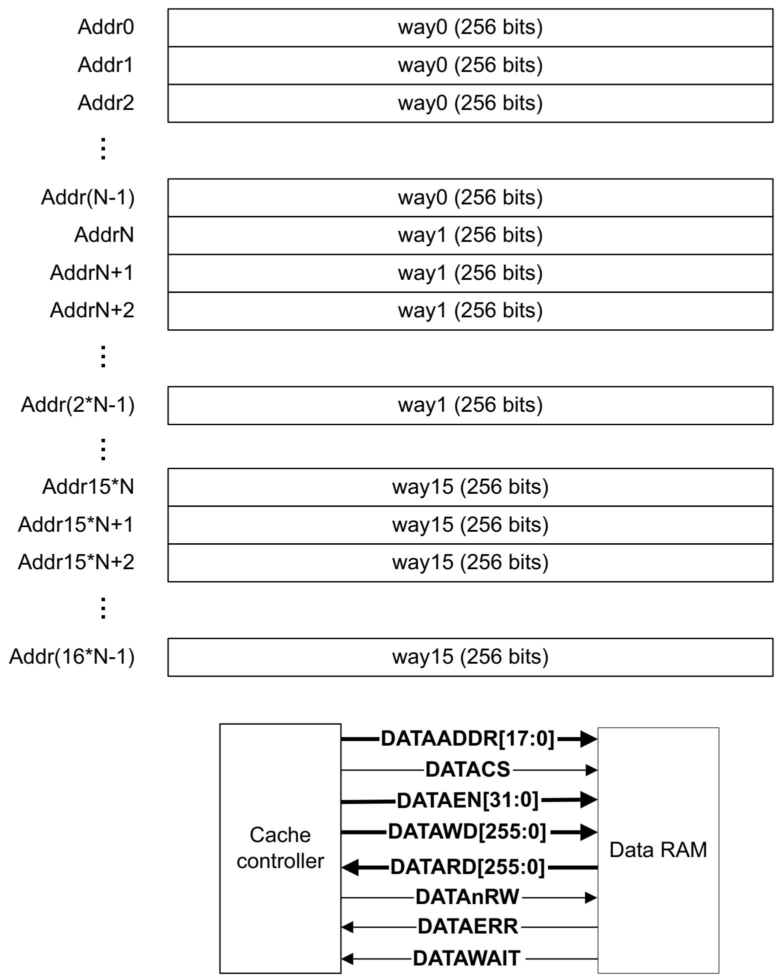


Figure 2-8 Multi-banking

Data RAM without banking

The data RAM shown in Figure 2-9 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.



L2 Cache Size	N =
256KB	512
512KB	1024
1MB	2048
2MB	4096
4MB	8192
8MB	16384

Note: This table is only applicable for a 16-way configuration.

Figure 2-9 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 cache controller samples the data or error when the Data RAM deasserts the **DATAWAIT** signal.

Data RAM with banking

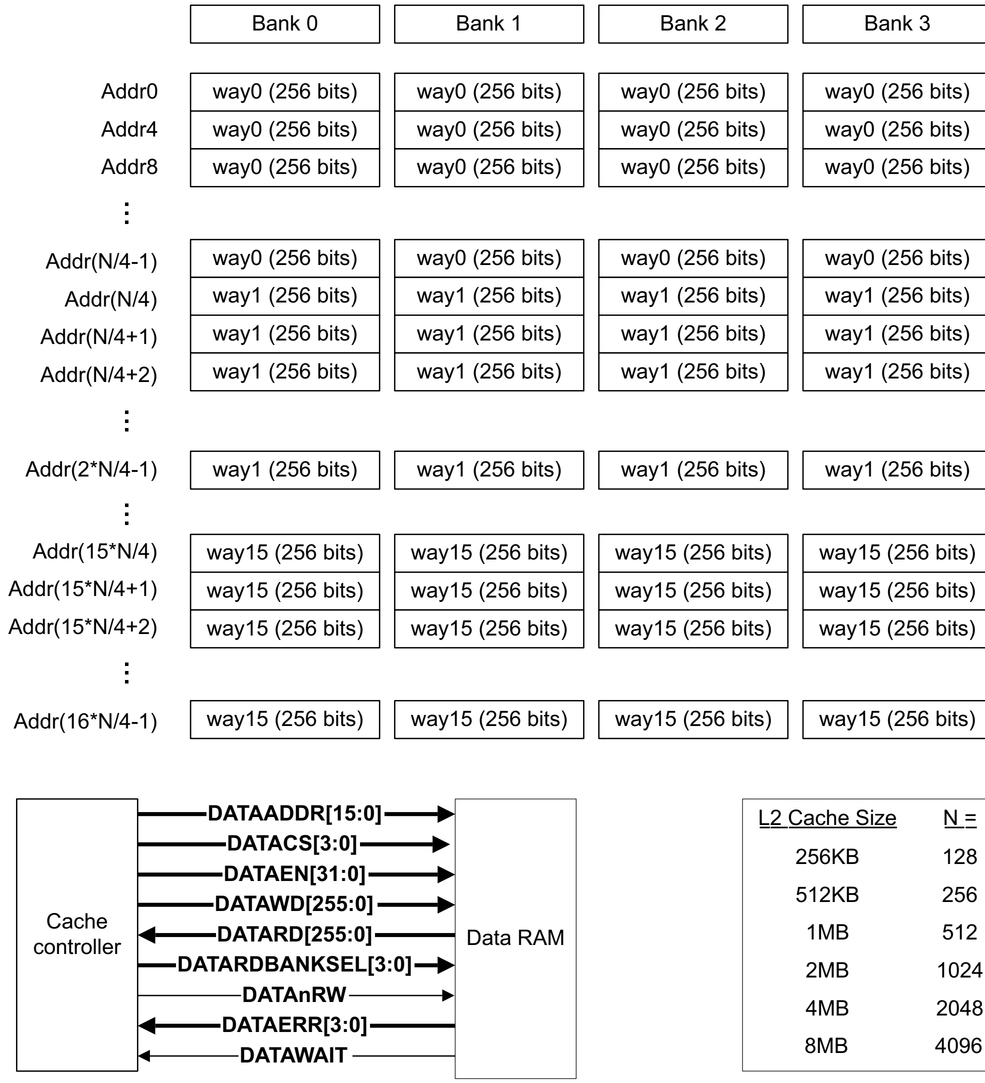
When you implement banking, the system partitions the data RAM into four banks. This enables pipelined streaming accesses. Table 2-17 shows how the system selects the banks through bits [6:5] of the address.

Table 2-17 Data RAM bank identification

AXI address [6:5]	Bank
0b00	0
0b01	1
0b10	2
0b11	3

Figure 2-10 on page 2-25 shows that the system has organized the data RAM as four banks of n-way 256-bit wide contiguous memories, where n is 8, or 16. This configuration supports the following accesses:

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



L2 Cache Size	N =
256KB	128
512KB	256
1MB	512
2MB	1024
4MB	2048
8MB	4096

Note: This table is only applicable for a 16-way configuration.

Figure 2-10 Data RAM with banking and 16 ways

Data parity RAM without banking

Figure 2-11 shows the required RAM organization for data parity RAM.

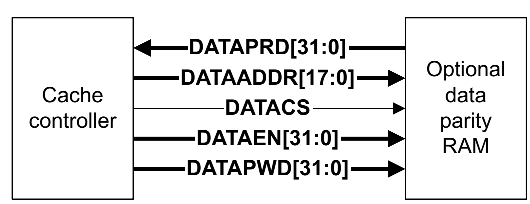


Figure 2-11 Data parity RAM without banking organization

Data parity RAM is always 32 bits wide, and connection is the same as for data RAM described in *Data RAM without banking* on page 2-23.

Data parity RAM with banking

Figure 2-12 shows the required RAM organization for data parity RAM with banking.

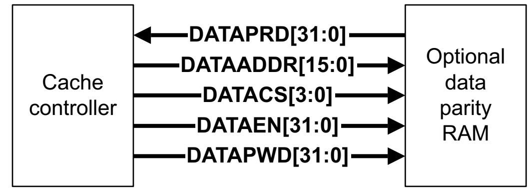


Figure 2-12 Data parity RAM with banking organization

Data parity RAM is always 32 bits wide, and connection is the same as for data RAM described in *Data RAM with banking* on page 2-24.

Tag RAM

Figure 2-13 on page 2-27 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 2-27.
- 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 non-secure data, and 0 for secure data.

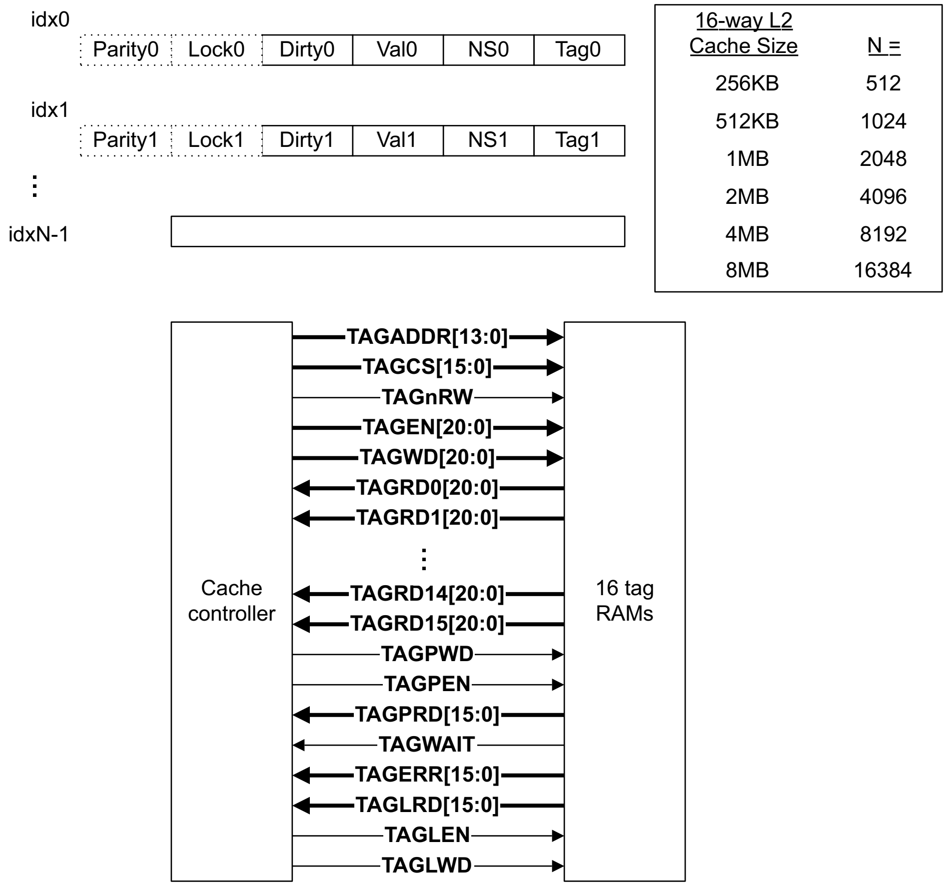


Figure 2-13 Tag RAM organization for a 16-way 256KB L2 cache, with parity, with lockdown by line

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 cache controller samples the tag signals or error signal when the tag ram deasserts the **TAGWAIT** signal.

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 **TAGPRD** signals to the additional read data bus bits and the **TAGPWD** signals to the additional write data bus bits. The tag parity RAMs can be split from the tag RAM. Additional logic is required for the tag parity RAM control signals.

Tag lock RAM

When you implement the lockdown by line feature, the simplest implementation of tag lock RAM is to have tag RAMs one bit wider and connect the respective **TAGLRD** signals to the additional read data bus bits and the **TAGLWD** signal to the additional write data bus bits. The tag lock RAMs can also be split from the tag RAM. Additional logic is required for the tag lock RAM control signals.

RAM bus usage versus cache associativity and way size

This section describes:

- Data RAM usage with or without banking

- Tag RAM usage.

Data RAM usage without banking

Figure 2-14 shows the **DATAADDR** bus format without banking:

17 14	13 0
Way	Index

Figure 2-14 Data RAM address bus format for 16 ways without banking

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 *Least Significant Bits* (LSB)s of the RAM address bus.
- 32KB way size:
  - bits [13:10] are left unconnected
  - bits [9:0] are the LSBs of the RAM address bus.
- 64KB way size:
  - bits [13:11] are left unconnected
  - bits [10:0] are the LSBs of the RAM address bus.
- 128KB way size:
  - bit [13:12] is left unconnected
  - bits [11:0] are the LSBs of the RAM address bus.
- 256KB way size:
  - bit [13] is left unconnected
  - bits [12:0] are the LSBs 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.

Data RAM usage with banking

Figure 2-15 shows the **DATAADDR** bus format with banking:

15 12	11 0
Way	Index

Figure 2-15 Data RAM address bus format for 16 ways with banking

Note

For 8 ways, the Way field is [14:12].

Bits [11:0] usage depends on the way size:

- 16KB way size:
  - bits [11:7] left unconnected
  - bits [6:0] are the LSBs of the RAM address bus.
- 32KB way size:
  - bits [11:8] are left unconnected
  - bits [7:0] are the LSBs of the RAM address bus.
- 64KB way size:
  - bits [11:9] are left unconnected
  - bits [8:0] are the LSBs of the RAM address bus.
- 128KB way size:
  - bit [11:10] is left unconnected
  - bits [9:0] are the LSBs of the RAM address bus.
- 256KB way size:
  - bit [11] is left unconnected
  - bits [10:0] are the LSBs of the RAM address bus.
- 512KB way size:
  - All [11:0] bits are used.

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

Tag RAM usage

TAGADDR, **TAGRDn**, and **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 2-32.

Note

Unless otherwise specified, the text and figures in this section apply to both tag RAM and data RAM.

RAM clocking

You can use clock enables if you run the RAMs 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 2-18 shows the RAM clock enables and their functions.

Table 2-18 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.

Table 2-18 RAM clock enables (continued)

Signal	Function
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. When you implement banking, four clock enable outputs exist, one for each bank.
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. When you implement banking, four clock outputs exist, one for each bank.

Figure 2-16 shows an example of the tag RAM running at half the frequency compared to the cache controller.

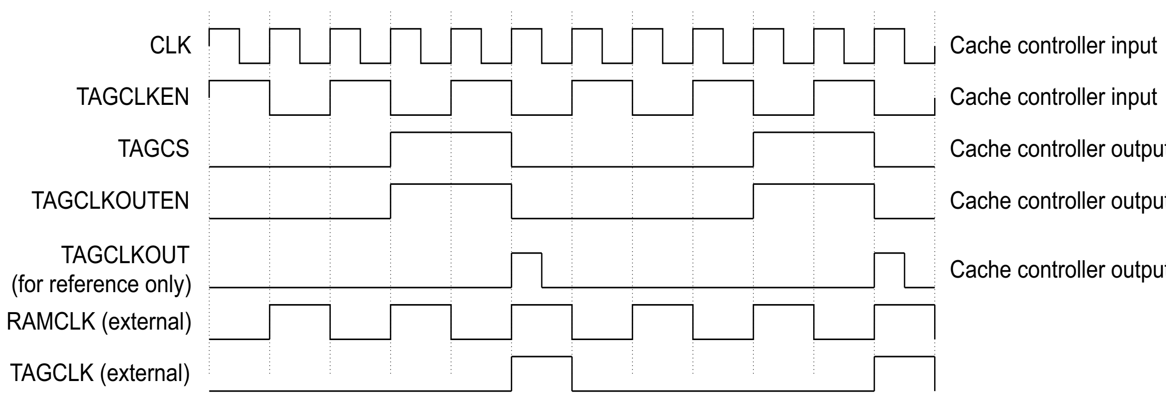


Figure 2-16 Tag RAM running at slower frequency

Figure 2-17 shows how the different clock gates used in the tag RAM clocking can be implemented.

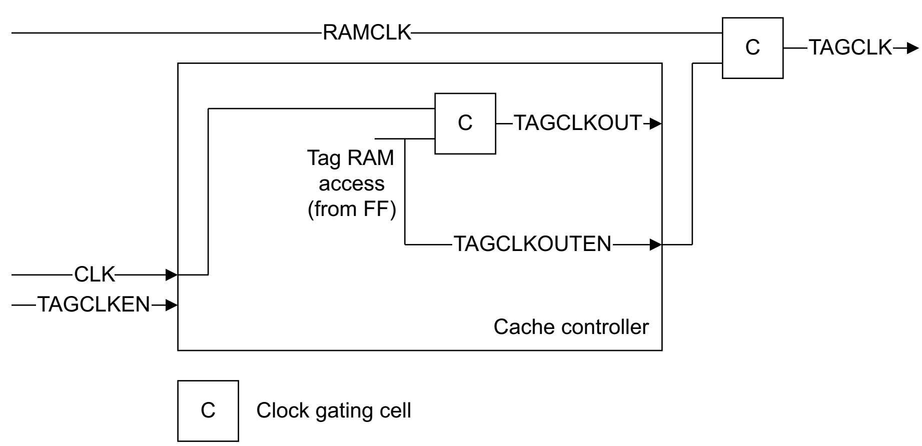


Figure 2-17 Tag RAM clock gating

Note

The implementer of the RAM array must also be the one that implements the clock gating cell that outputs **TAGCLK** in Figure 2-17.

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 3-12.

Setup latency is the number of cycles that the RAM control signals remain valid prior to the RAM clock edge. Figure 2-18 shows a timing diagram where the tag RAM setup latency has been programmed with the value `0x1`.

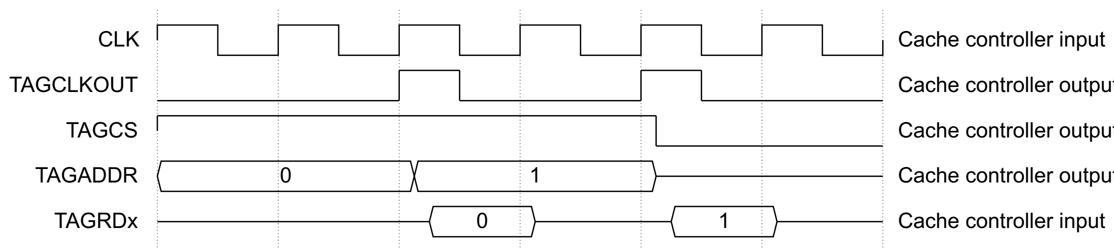


Figure 2-18 Tag RAM setup latency

Read access latency is the number of cycles taken by the read data to become valid after the RAM clock edge. Figure 2-19 shows a timing diagram where the tag RAM read access latency has been programmed with the value `0x1`.

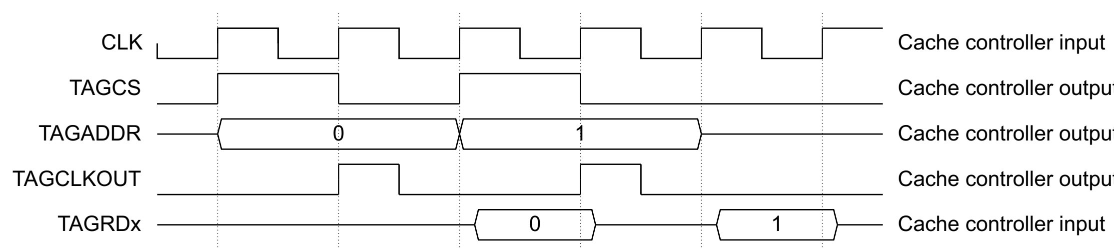


Figure 2-19 Tag RAM read access latency

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 2-20 shows a timing diagram where the tag RAM write access latency has been programmed with the value `0x1`.

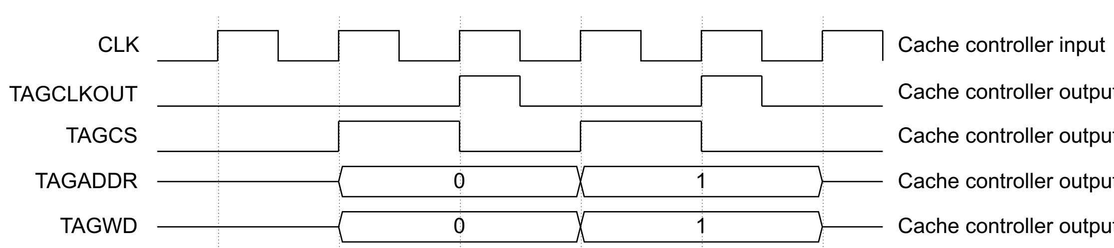


Figure 2-20 Tag RAM write access latency

The Data RAM interface behaves differently from the Tag RAM interface, especially regarding the **ADDR**, **EN**, **nRW**, and **WD** signals. While these signals remain stable during the setup latency by definition, you must not consider them stable for the following periods:

- after the setup latency time
- during the Data RAM read, write access latencies.

You must take this behavior into account when you synthesize the cache controller, especially if you implement multi-cycle paths on the Data RAM interface signals. Figure 2-21 shows the data RAM write access latency timing diagram.

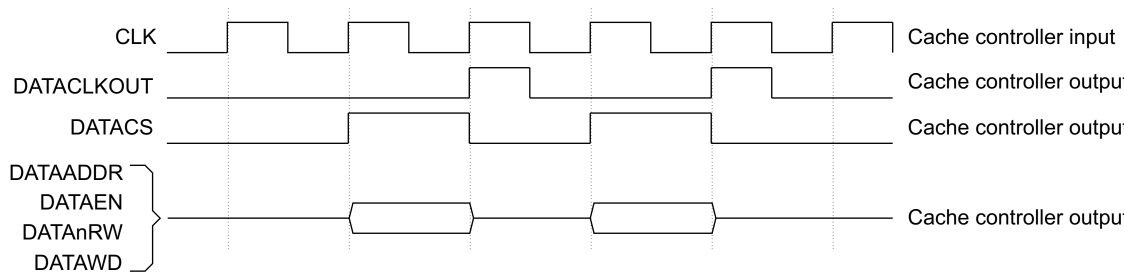


Figure 2-21 Data RAM write access latency

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.

When you implement banking for the data RAM, the programmed data RAM latencies apply to all four banks. Implement banking on the data RAM if you want to reduce overall access latency. Data RAM banking has no effect on the setup latency.

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 2-22 on page 2-34 shows the cache controller MBIST interface.

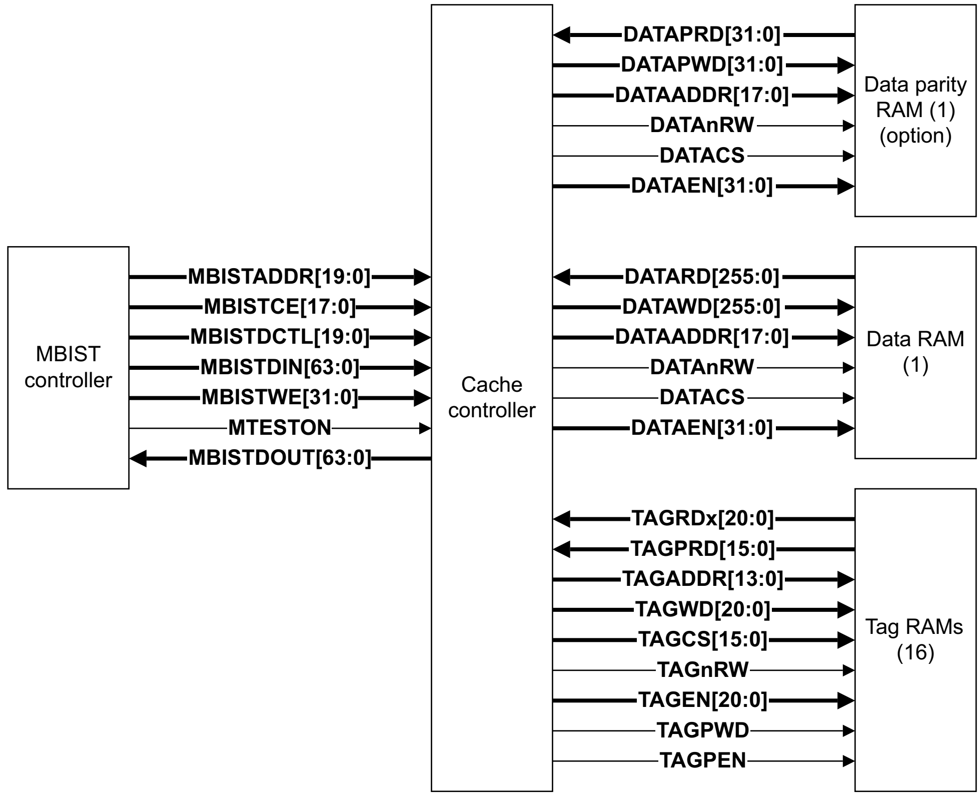


Figure 2-22 MBIST interface for 16-way implementation, with parity, without lockdown by line, without banking

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:

2.5.1 Reset requirement

You can assert **nRESET** LOW asynchronously to **CLK**. You must de-assert **nRESET** HIGH synchronously to **CLK**. To do this without over-constraining the reset timing, ARM recommends that you set a multi-cycle path of <n> clock cycles for the reset tree.

To reset the L2C-310, do the following:

1. Assert **nRESET** LOW asynchronously to **CLK**.
2. Wait for <m> clock cycles where <m> is greater than <n>.
3. Stop the clock **CLK**.
4. Wait for a period of <p> clock cycles, where the period is greater than the clock tree latency.
5. De-assert **nRESET** HIGH.
6. Wait for a period of <m> clock cycles.
7. Restart the clock **CLK**.
8. Wait for a minimum of two clock cycles before sending the first transaction to the L2C-310 AXI slave ports. This guarantees that the L2C-310 correctly samples its **CFGBIGEND**, **CFGADDRFILTEN**, **CFGADDRFILTEND**, and **CFGADDRFILTSTART** input signals.

2.5.2 Disabled operation

When the cache controller block is present but not enabled, transactions pass 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.

2.5.3 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 other potential 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 ports
- store buffer is drained at each strongly ordered write occurrence in slave ports
- as soon as all three slots of the store buffer contain data, the least recently accessed slot starts draining
- if a hazard is detected in a store buffer slot, that slot 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
- store buffer slots are automatically drained after 256 cycles of presence in the store buffer.

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.

2.5.4 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 that it is responsible for. 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 cache controller performs hazard checking on bits [31:5] of the address.

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.

2.5.5 Cortex-A9 optimizations

The L2C-310 implements several optimizations that the Cortex-A9 processor can use. These optimizations are:

- *Early write response*
- *Prefetch hints*
- *Full line of zero write*
- *Speculative reads of the Cortex-A9 MPCore processor* on page 2-38
- *Store buffer device limitation* on page 2-38.

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 L2C-310 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 HIGH the **Early BRESP enable** bit in the Auxiliary Control Register, bit[30]. The L2C-310 slave ports then send an early write response only if the input signal **AWUSERSx[11]** (where x=0 or 1) is set to 1 for the corresponding write transaction.

Prefetch hints

When you configure the Cortex-A9 processor to run in SMP mode, the automatic data prefetchers, implemented in one or more CPUs, issues special read accesses to the L2C-310. See the *Cortex-A9 TRM*. These special reads are called Prefetch Hints. They are indicated when a device sets **ARUSERSx[8]** to 1. When the L2C-310 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 for a miss. This behavior is not compatible with AXI. When a master other than a Cortex-A9 processor drives the L2C-310 **ARUSERSx[8]** must be tied 0.

Full line of zero write

When the L2C-310 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 L2C-310 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 L2C-310.

To enable this feature, perform the following steps:

1. enable `Full line of zero` feature in the L2C-310
2. turn on L2C-310
3. enable `Full line of zero` feature in A9.

Note

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

Speculative reads of the Cortex-A9 MPCore processor

An RTL configurable option supports the logic for speculative reads. This is only implemented in L2C-310 if the Verilog `` `define pl310_SPECULATIVE_READ `` is uncommented. When L2C-310 is connected to the Cortex-A9 MPCore processor, you must implement the logic. When L2C-310 is connected to another ARM CPU, and another AXI master, ARM recommends that you do not implement this logic because it permits removing multiple address comparators.

When you enable the speculative read feature, using a dedicated software control bit in the Cortex-A9 MPCore processor, on coherent linefills, the Cortex-A9 SCU speculatively issues read transactions to the L2C-310 in parallel with its Tag lookup. The SCU indicates these read transactions by setting **ARUSERSx[9]** HIGH. These transactions are not AXI-compliant because the L2C-310 does not return data on speculative reads. It prepares data in its Line Read Buffers. If the SCU misses, it issues a confirmation linefill to the L2C-310. The confirmation is a standard AXI transaction that is merged with the previous speculative read in the L2C-310 slave port. This enables the L2C-310 to return data to the level 1 cache sooner for a level 2 cache hit. If the SCU hits, the speculative read is naturally terminated in L2C-310, either after a certain number of cycles, or when a resource conflict exists. When a speculative read ends in the L2C-310 slave ports, either by confirmation or termination, the L2C-310 informs the Cortex-A9 SCU by asserting the **SRENDSx** and **SRIDSx** outputs. This represents the AXI ID of the terminated speculative read. When L2C-310 is not connected to the Cortex-A9 MPCore processor, you can leave the **SRENDSx** and **SRIDSx** outputs unconnected.

Note

You must only enable the speculative read feature when both clock enable inputs **INCLKEN**, and **OUTCLKEN** of the L2C-310 AXI slave interfaces are tied HIGH.

Store buffer device limitation

In some systems, write transactions to device memory regions can target slow peripherals. Because the L2C-310 store buffer is shared between all types of bufferable writes, a heavy traffic of Device, writes that progress slowly in the system can affect the performance of normal memory writes if these two types of traffic occur at the same time. To minimize this effect, you can limit the number of device writes in the L2C-310 store buffer so that at least one slot is always available for normal memory traffic. Bit 11 of the Auxiliary Control Register controls this feature.

2.5.6 Prefetching operation

The prefetch operation is the capability of attempting to fetch cache lines from L3 in advance, to improve system performance. The following sections describe the prefetch functionality:

- *Internal instruction and data prefetch engine* on page 2-39
- *Double linefill issuing* on page 2-39
- *Prefetch dropping* on page 2-41.

Internal instruction and data prefetch engine

To enable the prefetch feature, set bit 29 or 28 of the Auxiliary or Prefetch Control Register. When enabled, if one of the slave ports receives a cacheable read transaction, a cache lookup is performed on the subsequent cache line. Bits [4:0] of the Prefetch Control Register provide the address of the subsequent cache line. If a miss occurs, the cache line is fetched from L3, and allocated to the L2 cache.

By default, the prefetch offset is `0b00000`. 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 result in optimal performance. In some systems, it might be better to prefetch more in advance to achieve better performance. The prefetch offset enables 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. ARM recommends that you perform system experiments by varying the prefetch offset, to find the optimal value.

Note

The prefetch mechanism is not launched for a 4KB boundary crossing.

Double linefill issuing

The L2C-310 cache line length is 32-byte. Therefore, by default, on each L2 cache miss, L2C-310 issues 32-byte linefills, 4 x 64-bit read bursts, to the L3 memory system. L2C-310 can issue 64-byte linefills, 8 x 64-bit read bursts, on an L2 cache miss. When the L2C-310 is waiting for the data from L3, it performs a lookup on the second cache line targeted by the 64-byte linefill. If it misses, data corresponding to the second cache line are allocated to the L2 cache. If it hits, data corresponding to the second cache line are discarded.

You can control this feature using Bits 30, 27, and 23 of the Prefetch Control Register. Bit 23, and Bit 27 are only used if you set Bit 30 HIGH. Table 2-19 shows the behavior of the L2C-310 master ports, depending on the configuration you choose.

Table 2-19 L2C-310 master port behavior

Bit 30 Double linefill enable	Bit 27 Double linefill on WRAP read disable	Bit 23 Incr double Linefill enable	Original read address from L1	Read address to L3	AXI burst type	AXI burst length	Targeted cache lines
0	0 or 1	0 or 1	`0x00`	`0x00`	WRAP	`0x3`, 4x64-bit	`0x00`
0	0 or 1	0 or 1	`0x20`	`0x20`	WRAP	`0x3`, 4x64-bit	`0x20`
1	0 or 1	0	`0x00`	`0x00`	WRAP	`0x7`, 8x64-bit	`0x00` and `0x20`
1	1	0	`0x08` or `0x10` or `0x18`	`0x08` or `0x10` or `0x18`	WRAP	`0x3`, 4x64-bit	`0x00`
1	0	0	`0x08` or `0x10` or `0x18`	`0x00`	WRAP	`0x7`, 8x64-bit	`0x00` and `0x20`
1	0 or 1	0	`0x20`	`0x20`	WRAP	`0x7`, 8x64-bit	`0x00` and `0x20`

Table 2-19 L2C-310 master port behavior (continued)

Bit 30 Double linefill enable	Bit 27 Double linefill on WRAP read disable	Bit 23 Incr double Linefill enable	Original read address from L1	Read address to L3	AXI burst type	AXI burst length	Targeted cache lines
1	1	0	`0x28` or `0x30` or `0x38`	`0x28` or `0x30` or `0x38`	WRAP	`0x3`, 4x64-bit	`0x20`
1	0	0	`0x28` or `0x30` or `0x38`	`0x20`	WRAP	`0x7`, 8x64-bit	`0x00` and `0x20`
1	0 or 1	1	`0x00`	`0x00`	WRAP	`0x7`, 8x64-bit	`0x00` and `0x20`
1	1	1	`0x08` or `0x10` or `0x18`	`0x08` or `0x10` or `0x18`	WRAP	`0x3`, 4x64-bit	`0x00`
1	0	1	`0x08` or `0x10` or `0x18`	`0x00`	WRAP	`0x7`, 8x64-bit	`0x00` and `0x20`
1	0 or 1	1	`0x20`	`0x20`	INCR	`0x7`, 8x64-bit	`0x20` and `0x40`
1	1	1	`0x28` or `0x30` or `0x38`	`0x28` or `0x30` or `0x38`	WRAP	`0x3`, 4x64-bit	`0x20`
1	0	1	`0x28` or `0x30` or `0x38`	`0x20`	INCR	`0x7`, 8x64-bit	`0x20` and `0x40`

Note

- Double linefills are not issued for prefetch reads if you enable exclusive cache configuration
- Double linefills are not launched when crossing a 4KB boundary.
- Double linefills only occur if a WRAP4 or an INCR4 64-bit transaction is received on the slave ports. This transaction is most commonly seen as a result of a cache linefill in a master, but can be produced by a master when accessing memory marked as inner non-cacheable.
- When bit 27 is at its default value of 0, Table 2-19 on page 2-39 shows that critical word first accesses are not always maintained between the original read from L1 and the actual read to L3.
- The following is the most likely configuration to achieve optimal performance for the double linefill feature:
  - Bit 30 = 1
  - Bit 27 = 1 where the critical word first is always maintained, and there is double linefill only on reads aligned on cache line boundaries

— Bit 23 = 1 where the double linefill feature prefetches next cache line.

Prefetch dropping

L2C-310 can operate with the following types of prefetch accesses:

- prefetch hints received from the Cortex-A9 MPCore processor
- internally generated prefetch transactions.

Prefetch accesses can use a large amount of the address slots present in the L2C-310 master ports. This prevents non-prefetch accesses being serviced, and affects performance. To counter this effect, L2C-310 can drop prefetch accesses. You can control this using bit 24 of the Prefetch Control Register. See *Prefetch Control Register* on page 3-34.

When enabled, if a resource conflict exists between prefetch and non-prefetch accesses in the L2C-310 master ports, prefetch accesses are dropped. When data corresponding to these dropped prefetch accesses return from L3, they are discarded, and not allocated into the L2 cache.

2.5.7 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.

ARM recommends 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
Non-cacheable 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.

2.5.8 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 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.

Table 2-21 Event pins (continued)

Pin	Description
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.
EPFALLOC	Prefetch hint allocated into the L2 cache.
EPFHIT	Prefetch hint hits in the L2 cache.
EPFRCVDS0	Prefetch hint received by slave port S0.
EPFRCVDS1	Prefetch hint received by slave port S1.
IPFALLOC	Allocation of a prefetch generated by L2C-310 into the L2 cache.
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.
SRCONFS0	Speculative read confirmed in slave port S0.
SRCONFS1	Speculative read confirmed in slave port S1.
SRRCVDS0	Speculative read received by slave port S0.
SRRCVDS1	Speculative read received by slave port S1.
WA	Allocation into the L2 cache caused by a write, with Write-Allocate attribute, miss.

2.5.9 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 tag RAM write
PARRDINTR	Parity error on L2 data RAM read

Table 2-22 Interrupts (continued)

Pin	Description
PARRTINTR	Parity error on L2 tag RAM read
ECNTRINTR	Event Counter Overflow or Event Counter Increment
L2CCINTR	L2CC Combined Interrupt Output

See *Interrupt registers* on page 3-17.

2.5.10 Parity and RAM error support

Figure 2-23 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 the controller writes to only one tag RAM at any one time, it only requires one bit.

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.

Figure 2-23 shows the cache controller parity and RAM error support.

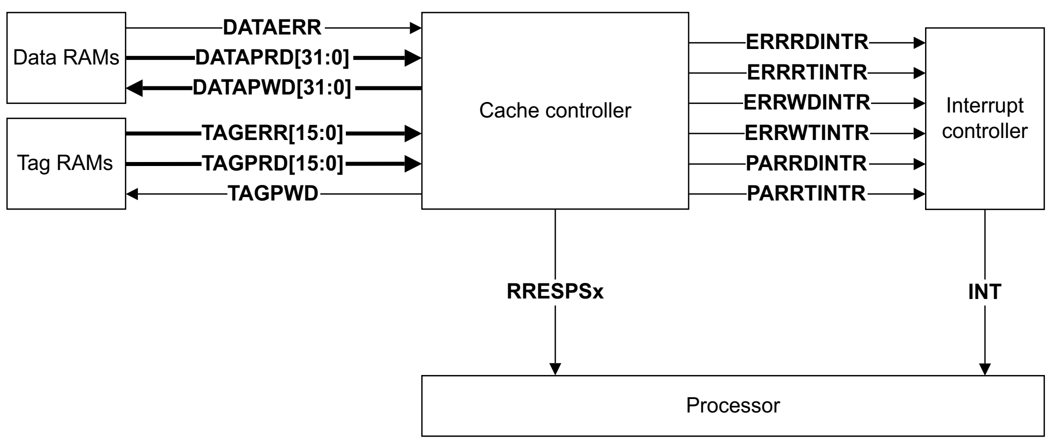


Figure 2-23 Parity and RAM error support for a 16-way implementation without banking

The L2C-310 can report parity and RAM errors to the processor through the interrupt lines and also the **RRESPSx** signal 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 2-45
- *Write access from the store buffer* on page 2-45
- *AXI master or store buffer allocation requests* on page 2-45
- *Clean maintenance* on page 2-46
- *Invalidate maintenance operation* on page 2-46
- *Clean and Invalidate maintenance operation* on page 2-46.

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 lookup	The controller flags an error through the **PARRTINTR** or the **ERRRTINTR** interrupt signal. It reports SLVERR through the **RRESPSx** signal.
Data RAM error on data read in case of hit	The controller flags an error through the **PARRDINTR** or the **ERRRDINTR** interrupt signal. It reports SLVERR through the **RRESPSx** signal.

Write access from the store buffer

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

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

Case	Effect
Tag parity or RAM error on tag read	The controller flags an error through the **PARRTINTR** or the **ERRRTINTR** interrupt signal. No attempt is made to write the cache.
Data RAM error on data write	The controller flags an error through the **ERRWDINTR** interrupt signal. Subsequent reads to the same line 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 a tag read, as part of an eviction	The controller flags an error through the **PARRTINTR** or the **ERRRTINTR** interrupt signal. It masks the eviction and issues a write to the L3 memory system with all strobes set to 0. It does not stop the allocation process.
Data parity or RAM error on a data read for eviction	The controller flags an error through the **PARRDINTR** or the **ERRRDINTR** interrupt signal. It does not stop the allocation process. For a data RAM error the controller masks the eviction and issues a write transaction to the L3 memory system with all strobes set to 0. For a data parity error the controller: • does not mask the eviction • sends the corrupted data to the L3 memory system.
Tag RAM error on a tag write	The controller flags an error through the **ERRWTINTR** interrupt signal. Subsequent cache lookups to the same index are unpredictable.
Data RAM error on a data write	The controller flags an error 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	The controller flags an error through the **PARRTINTR** or the **ERRRTINTR** interrupt signal. It cancels the clean operation.
Data parity or RAM error on data read	The controller flags an error through the **PARRDINTR** or the **ERRRDINTR** interrupt signal. For a data RAM error the controller masks the eviction that results from the clean operation. It issues a write to the L3 memory system with all strobes set to 0. For a data parity error the controller does not mask the eviction. It sends the corrupted data to the L3 memory system.

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	The controller flags an error through the **PARRTINTR** or the **ERRTRINTR** interrupt signal. It cancels the invalidate operation.
Tag RAM error on valid information write	The controller flags an error 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	The controller flags an error through the **PARRTINTR** or the **ERRRTINTR** interrupt signal. The operation is canceled.
Data parity or RAM error on data read	The controller flags an error through the **PARRTINR** or the **ERRRDINTR** interrupt signal. For a data RAM error the controller masks the eviction resulting from the operation. It issues a write to the L3 memory system with all strobes set to 0. For a data parity error the controller does not mask the eviction resulting from the operation. It sends the corrupted data to the L3 memory system.
Tag RAM error on valid information write	The controller flags an error 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*
- *Dynamic clock gating*
- *Standby mode*
- *Dormant mode*
- *Shutdown mode* on page 2-48.

2.6.1 Run mode

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

2.6.2 Dynamic clock gating

When you enable the dynamic high-level clock-gating feature, the cache controller stops its clock when it is idle. It does not stop the clock immediately when it is idle, but after it counts several clock cycles. See *Power Control Register* on page 3-36. When it stops the clock, it drives the **READY** outputs of the AXI slave interfaces LOW and the **CLKSTOPPED** output HIGH. If one of its interfaces detects an AXI transaction, it restarts its clock and asserts its **READY** signals HIGH to accept the new transaction.

2.6.3 Standby mode

You can use the standby mode of the L2C-310 in conjunction with the `Wait For Interrupt` mode of the processor that drives the L2C-310. When a processor is in the `Wait For Interrupt` mode, the usual protocol is to set an output HIGH. For example, the Cortex-A9 MPCore processor sets its **SCUIDLE** output HIGH to indicate that it is in `Wait For Interrupt` mode. For this example, you must connect **SCUIDLE** to the **STOPCLK** input of the L2C-310. When **STOPCLK** is HIGH, and the standby mode feature is enabled, the L2C-310 does the following:

1. Waits for the IDLE state.
2. Drives the **READY** outputs of its slave interfaces LOW.
3. Asserts the **CLKSTOPPED** output HIGH.
4. Stops its clock.

See *Power Control Register* on page 3-36.

2.6.4 Dormant mode

Dormant mode enables you to power-down the cache controller and leave the cache memories powered-up so that they maintain their state. Dormant mode prevents standard cells, that are used to implement the cache controller, from using power.

Because the internal configuration registers are implemented in standard cells, you must save their value before you can remove power from the cache controller. You must therefore perform the following steps when you enter and exit dormant mode:

1. Save all cache controller configuration registers.
2. Perform a cache sync operation.
3. Put the cache controller in standby mode.
4. Wait for **CLKSTOPPED** to be asserted.
5. Remove the power.
6. Restore the power, and reset the cache controller.

7. Restore the configuration registers.

As with standby mode, it is expected that the cache controller enters dormant mode only when the L1 masters are inactive, and issue no additional transactions. The external power controller triggers the transition from Dormant mode to Run mode. The external power controller asserts the reset. Ensure the cache controller is placed back into run mode prior to the L1 masters.

In Dormant mode, you must keep the RAMs powered-up while the cache controller is powered-down. Because of this, you must implement clamping cells 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.

2.6.5 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:

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.
- You must preserve the reserved bits in all registers otherwise unpredictable behavior of the device might occur.
- 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 permitted, otherwise a SLVERR response is returned.
- Exclusive accesses are not permitted. 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, Data RAM Latency, Prefetch, and Power Control registers using a read-modify-write to set up global configurations:
   - associativity, Way Size
   - latencies for RAM accesses
   - allocation policy
   - prefetch and power capabilities.

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 and *Secure* (S)
0x100 - 0x1FC	Control	Control	Write S Read NS and S
0x200 - 0x2FC	Interrupt and Counter Control Registers	Interrupt and Counter Control Registers	NS and 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 and S
0xD00 - 0xEFC	Reserved	Reserved	-
0xF00 - 0xFFC	Debug, Prefetch and Power	Debug, Prefetch and Power	Write S Read NS and S

All register addresses in the cache controller are fixed relative to the base address. Table 3-2 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	0x410000C9[a]	32	*Cache ID Register* on page 3-7
0x004	reg0_cache_type	RO	0x1C100100[b]	32	*Cache Type Register* on page 3-7
0x100	reg1_control	RW	0x00000000	32	*Control Register* on page 3-9
0x104	reg1_aux_control	RW	0x02020000[b]	32	*Auxiliary Control Register* on page 3-10
0x108	reg1_tag_ram_control	RW	0x00000nnn[c]	32	*Tag and Data RAM Latency Control Registers* on page 3-12
0x10C	reg1_data_ram_control	RW	0x00000nnn[d]	32	
0x200	reg2_ev_counter_ctrl	RW	0x00000000	32	*Event Counter Control Register* on page 3-14
0x204	reg2_ev_counter1_cfg	RW	0x00000000	32	*Event Counter Configuration Registers* on page 3-15
0x208	reg2_ev_counter0_cfg	RW	0x00000000	32	
0x20C	reg2_ev_counter1	RW	0x00000000	32	*Event counter value registers* on page 3-16
0x210	reg2_ev_counter0	RW	0x00000000	32	

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

Offset	Name	Type	Reset	Width	Description
0x214	reg2_int_mask[e]	RW	0x00000000	32	*Interrupt registers* on page 3-17
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 3-21
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	
0x900	reg9_d_lockdown0	RW	0x00000000	32	*Cache lockdown* on page 3-27
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	
0x954	reg9_unlock_way[g]	RW	0x00000000	32	-

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

Offset	Name	Type	Reset	Width	Description
`0xC00`	reg12_addr_filtering_start	RW	`0x00000000`[h]	32	*Address filtering* on page 3-32
`0xC04`	reg12_addr_filtering_end	RW	`0x00000000`[i]	32	
`0xF40`	reg15_debug_ctrl	RW	`0x00000000`	32	*Debug Register* on page 3-33
`0xF60`	reg15_prefetch_ctrl	RW	`0x00000000`	32	*Prefetch Control Register* on page 3-34
`0xF80`	reg15_power_ctrl	RW	`0x00000000`	32	*Power Control Register* on page 3-36

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

b. This value is pin dependent, depending on how external WAYSIZE 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 **CFGADDRFILTSTART** pins are tied.

i. This value is pin dependent if address filtering is implemented, depending on how external **CFGADDRFILTEND** pins are tied.

3.3 Register descriptions

This section describes the cache controller registers. Table 3-2 on page 3-4 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 3-4.

Figure 3-1 shows the reg0_cache_id Register bit assignments.

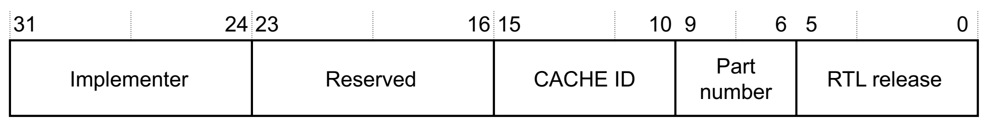


Figure 3-1 reg0_cache_id Register bit assignments

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	0x9

Note

- Part number 0x3 denotes CoreLink Level 2 Cache Controller L2C-310.
- RTL release 0x9 denotes r3p3 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 3-4.

Figure 3-2 shows the reg0_cache_type register bit assignments.

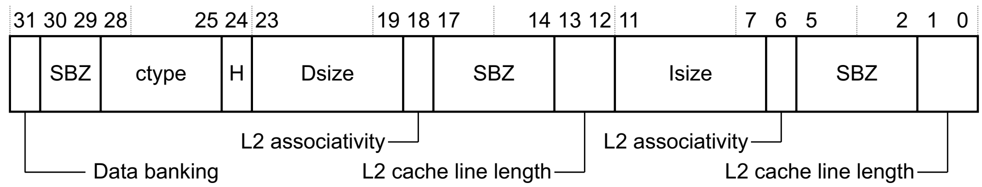


Figure 3-2 reg0_cache_type Register bit assignments

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]	Data banking	-	`0` data banking not implemented `1` data banking implemented
[30:29]	SBZ	-	`0b00`
[28:25]	ctype	-	`0b11`*xy*, where: *x*=1 if `pl310_LOCKDOWN_BY_MASTER` is defined, otherwise 0 *y*=1 if `pl310_LOCKDOWN_BY_LINE` is defined, otherwise 0. See *Cache lockdown* on page 3-27.
[24]	H	-	`0` unified `1` Harvard
[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]
[17:14]	SBZ	-	`0`
[13:12]	L2 cache line length	-	`0b00` 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	-	`0b00` 32 bytes

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 3-4.

Figure 3-3 shows the reg1_control Register bit assignments.

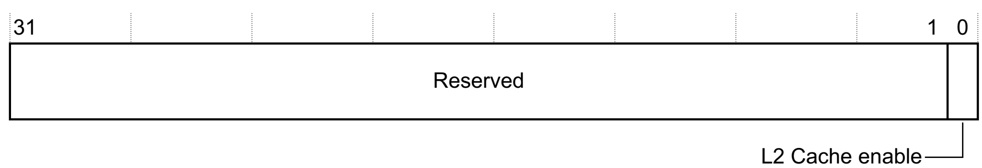


Figure 3-3 reg1_control Register bit assignments

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 with its reserved bits preserved. You can read it 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 the SLVERR response.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-4.

Figure 3-4 shows the reg1_aux_control Register bit assignments.

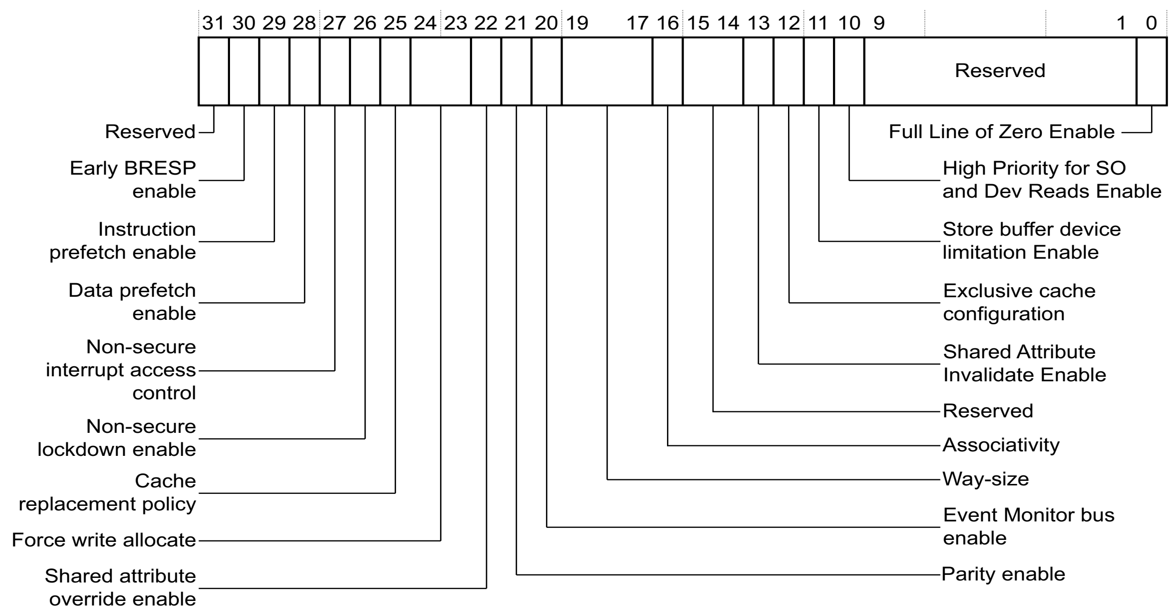


Figure 3-4 reg1_aux_control Register bit assignments

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 *Early write response* on page 2-37.
[29]	Instruction prefetch enable	`0`	Instruction prefetching disabled. This is the default.
		`1`	Instruction prefetching enabled.
		See *Prefetch Control Register* on page 3-34.	
[28]	Data prefetch enable	`0`	Data prefetching disabled. This is the default.
		`1`	Data prefetching enabled.
		See *Prefetch Control Register* on page 3-34.	
[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]	Cache replacement policy	`0`	Pseudo-random replacement using lfsr.
		`1`	Round-robin replacement. This is the default.
		See *Replacement strategy* on page 3-31.	
[24:23]	Force write allocate	`0b00`	Use **AWCACHE** attributes for WA. This is the default.
		`0b01`	Force no allocate, set WA bit always 0.
		`0b10`	Override **AWCACHE** attributes, set WA bit always 1, all cacheable write misses become write allocated.
		`0b11`	Internally mapped to 00. See *Cache operation* on page 2-11 for more information.
[22]	Shared attribute override enable	`0`	Treats shared accesses as specified in *Shareable attribute* on page 2-15. 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.
[19:17]	Way-size[a]	`0b000`	Reserved, internally mapped to 16KB.
		`0b001`	16KB
		`0b010`	32KB
		`0b011`	64KB
		`0b100`	128KB
		`0b101`	256KB
		`0b110`	512KB
		`0b111`	Reserved, internally mapped to 512 KB.

Table 3-6 reg1_aux_control Register bit assignments (continued)

Bits	Field	Description	
[16]	Associativity[b]	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 *Shareable attribute* on page 2-15.
[12]	Exclusive cache configuration	0	Disabled. This is the default.
		1	Enabled. See *Exclusive cache configuration* on page 2-17.
[11]	Store buffer device limitation Enable	0	Store buffer device limitation disabled. Device writes can take all slots in store buffer. This is the default.
		1	Store buffer device limitation enabled. Device writes cannot take all slots in store buffer when connected to the Cortex-A9 MPCore processor. There is always one available slot to service Normal Memory.
[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 L2C-310 master ports. This is the default.
		1	Strongly Ordered and Device reads get the highest priority when arbitrated in the L2C-310 master ports.
[9:1]	Reserved	SBZ/RAZ	
[0]	Full Line of Zero Enable	0	Full line of write zero behavior disabled. This is the default.
		1	Full line of write zero behavior Enabled.
		See *Full line of zero write* on page 2-37.	

a. The default value of the way size depends on how the external WAYSIZE pins are tied.

b. 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 3-4.

Figure 3-5 on page 3-13 shows the reg1_tag_ram_control and reg1_data_ram_control Register bit assignments.

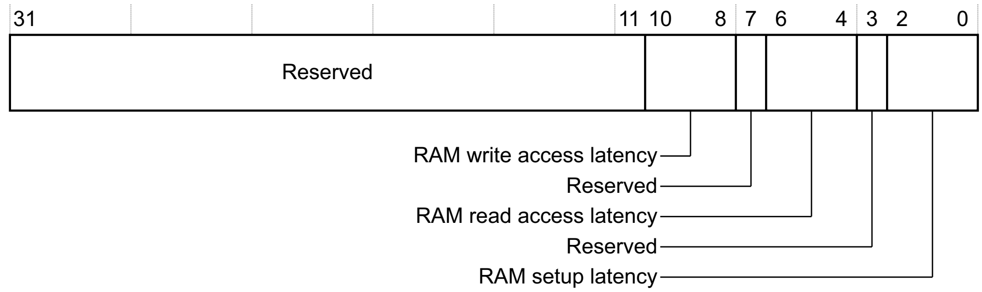


Figure 3-5 reg1_tag_ram_control and reg1_data_ram_control Register bit assignments

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. `0b000` 1 cycle of latency, there is no additional latency `0b001` 2 cycles of latency `0b010` 3 cycles of latency `0b011` 4 cycles of latency `0b100` 5 cycles of latency `0b101` 6 cycles of latency `0b110` 7 cycles of latency `0b111` 8 cycles of latency.
[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. `0b000` 1 cycle of latency, there is no additional latency `0b001` 2 cycles of latency `0b010` 3 cycles of latency `0b011` 4 cycles of latency `0b100` 5 cycles of latency `0b101` 6 cycles of latency `0b110` 7 cycles of latency `0b111` 8 cycles of latency.
[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. `0b000` 1 cycle of latency, there is no additional latency `0b001` 2 cycles of latency `0b010` 3 cycles of latency `0b011` 4 cycles of latency `0b100` 5 cycles of latency `0b101` 6 cycles of latency `0b110` 7 cycles of latency `0b111` 8 cycles of latency.

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 3-4.

Figure 3-6 shows the reg2_ev_counter_ctrl Register bit assignments.

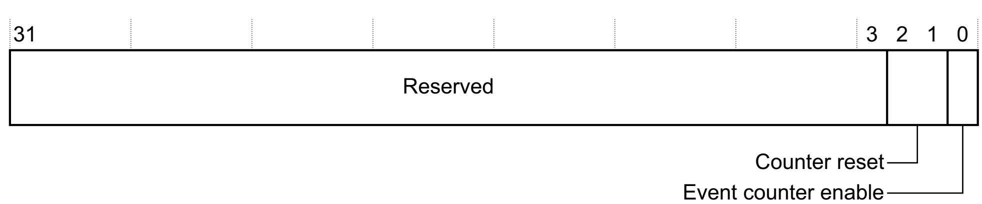


Figure 3-6 reg2_ev_counter_ctrl Register bit assignments

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. The following counters are reset when a 1 is written to the following bits: • 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 2-42 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 3-4.

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

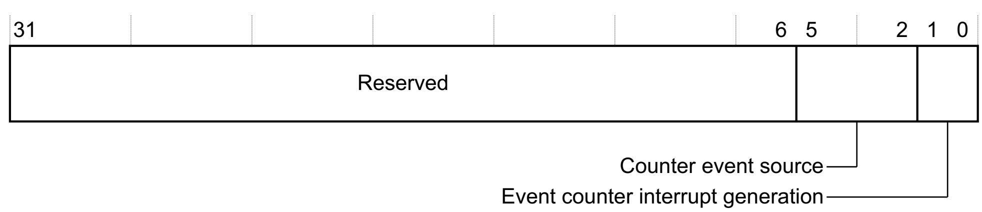


Figure 3-7 reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments

Table 3-9 shows the reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments.

Table 3-9 reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments

Bits	Field	Description
[31:6]	Reserved	SBZ/RAZ

Table 3-9 reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments

Bits	Field	Description	
[5:2]	Counter event source	Event	Encoding
		Counter Disabled	`0b0000`
		CO	`0b0001`
		DRHIT	`0b0010`
		DRREQ	`0b0011`
		DWHIT	`0b0100`
		DWREQ	`0b0101`
		DWTREQ	`0b0110`
		IRHIT	`0b0111`
		IRREQ	`0b1000`
		WA	`0b1001`
		IPFALLOC	`0b1010`
		EPFHIT	`0b1011`
		EPFALLOC	`0b1100`
		SRRCVD	`0b1101`
		SRCONF	`0b1110`
		EPFRCVD	`0b1111`
[1:0]	Event counter interrupt generation	`0b00`	Disabled. This is the default.
		`0b01`	Enabled: Increment condition.
		`0b10`	Enabled: Overflow condition.
		`0b11`	Interrupt generation is disabled.

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 3-4.

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. If a counter reaches its maximum value, it saturates at that value until it is reset.

3.3.9 Interrupt registers

The following interrupt registers exist:

- *Interrupt Mask Register* on page 3-18
- *Masked Interrupt Status Register* on page 3-19
- *Raw Interrupt Status Register* on page 3-20
- *Interrupt Clear Register* on page 3-21.

Figure 3-8 shows the register bit assignments.

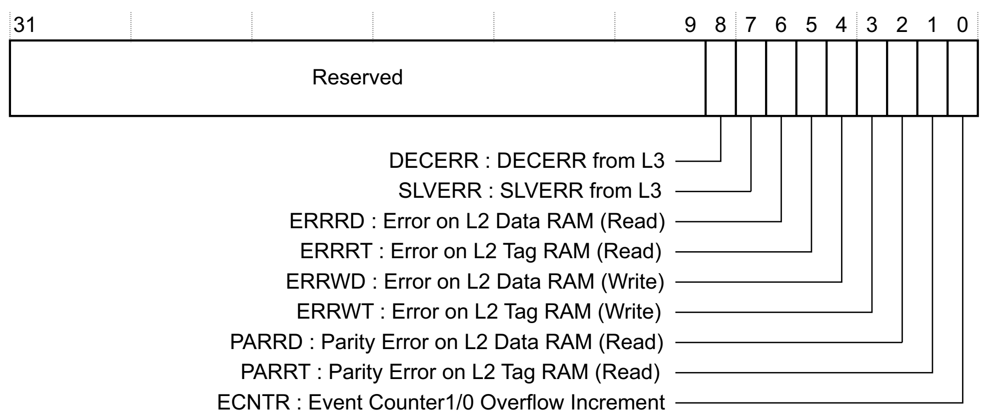


Figure 3-8 Interrupt Register bit assignments

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 3-17 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 0, 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 3-4.

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	`0`	Masked. This is the default.
[7]	SLVERR: SLVERR from L3	`1`	Enabled.
[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 and Event Counter 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 3-4.

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: **HIGH** If the bits read HIGH, they reflect the status of the input lines triggering an interrupt. **LOW** If the bits read LOW, either no interrupt has been generated, or the interrupt is masked.
[7]	SLVERR: SLVERR from L3	
[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 and Event Counter 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 3-4.

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	Bits read can be HIGH or LOW: **HIGH** If the bits read HIGH, they reflect the status of the input lines triggering an interrupt. **LOW** If the bits read LOW, no interrupt has been generated.
[7]	SLVERR: SLVERR from L3	
[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 and Event Counter0 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 0, 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 3-4.

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. When a bit is written as 0, it has no effect.
[7]	SLVERR: SLVERR from L3	
[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 and Event Counter0 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. To perform the maintenance operation, perform a write to the corresponding register. If the operation is specific to the Way or Set/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 3-24.

Table 3-15 Maintenance operations

Operation	Base offset	Type	Bit assignment format
Cache Sync	`0x730`	RW	See Figure 3-11 on page 3-23
Invalidate Line by PA	`0x770`	RW	See Figure 3-9
Invalidate by Way	`0x77C`	RW	See Figure 3-12 on page 3-23
Clean Line by PA	`0x7B0`	RW	See Figure 3-9
Clean Line by Set/Way	`0x7B8`	RW	See Figure 3-10
Clean by Way	`0x7BC`	RW	See Figure 3-12 on page 3-23
Clean and Invalidate Line by PA	`0x7F0`	RW	See Figure 3-9
Clean and Invalidate Line by Set/Way	`0x7F8`	RW	See Figure 3-10
Clean and Invalidate by Way	`0x7FC`	RW	See Figure 3-12 on page 3-23

Figure 3-9 shows the PA format.

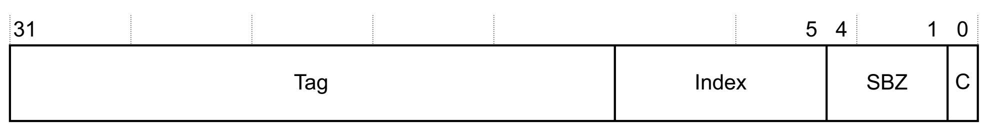


Figure 3-9 Physical address format

Note

The bit position of the boundary between the Tag field and the Index field varies according to the Index bit width.

Figure 3-10 shows the Index or Way format.

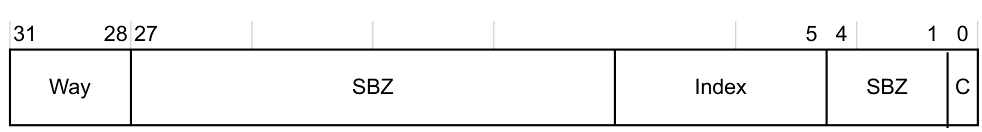


Figure 3-10 Index or way format

Note

The bit position of the boundary between the SBZ field and the Index field varies according to the Index bit width. For a 16-way implementation, all four bits [31:28] are used. If the 16-way option is not enabled, bit [31] is reserved.

Figure 3-11 shows the cache sync format.

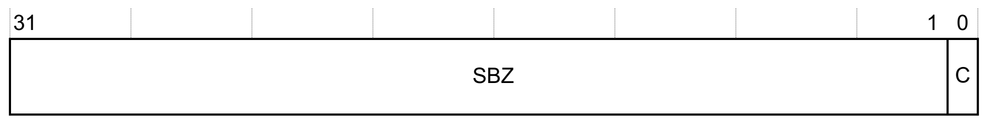


Figure 3-11 Cache sync format

Atomic operations

The following are atomic operations:

- Clean Line by PA or by Set/Way
- Invalidate Line by PA
- Clean and Invalidate Line by PA or by Set/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 3-12 shows the Way Format. You can select multiple ways at the same time, by setting the Way bits to 1.

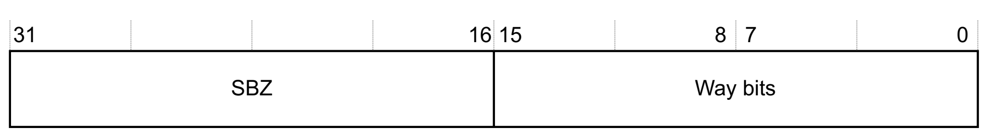


Figure 3-12 Way format

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 cache controller considers the targeted ways locked until it treats them. Therefore the controller does not permit read or write misses to access that way. It does however permit read or write hits to access the way, therefore there can still be dirty lines at the end of a 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, or ways, 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 Set/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, or ways, 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 Set/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, or ways, 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 L2C-310 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. Multiple masters have the ability to launch L2 cache maintenance operations. ARM recommends that you control access to the corresponding PL310 registers by using semaphores.

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* 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*, for example:

```
InvalLevel1 Address   ; Invalidate line in L1 cache
DSB                   ; Ensure completion of the L1 inval
InvalLevel2 Address   ; Invalidate line in L2 cache
```

```
CACHE SYNC           ; Ensure 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
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 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.

This is a problem for data but not for instruction in exclusive cache. However, if data cannot be prefetched, then there is no issue.

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 3-25. 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.

The following lockdown schemes exist:

- *Cache lockdown by line*
- *Cache lockdown by way* on page 3-29.

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.
- Unlock All Lines Register. See Table 3-19.

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 enabled.

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 3-31. For these tables each bit has the following meaning:

0 allocation can occur in the corresponding way.

1 there is no allocation in the corresponding way.

For the USER signals in these tables, y = R or W, and x = 0 or 1.

See also *Lockdown by way* on page 2-18.

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK000	Use when **AyUSERSx[7:5]** = `0b000`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK000	Use when **AyUSERSx[7:5]** = `0b000`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK001	Use when **AyUSERSx[7:5]** = `0b001`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK001	Use when **AyUSERSx[7:5]** = `0b001`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK010	Use when **AyUSERSx[7:5]** = `0b010`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK010	Use when **AyUSERSx[7:5]** = `0b010`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK011	Use when **AyUSERSx[7:5]** = `0b011`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK011	Use when **AyUSERSx[7:5]** = `0b011`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK100	Use when **AyUSERSx[7:5]** = `0b100`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK100	Use when **AyUSERSx[7:5]** = `0b100`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK101	Use when **AyUSERSx[7:5]** = `0b101`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK101	Use when **AyUSERSx[7:5]** = `0b101`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK110	Use when **AyUSERSx[7:5]** = `0b110`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK110	Use when **AyUSERSx[7:5]** = `0b110`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	DATALOCK111	Use when **AyUSERSx[7:5]** = `0b111`

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

Bits	Field	Description
[31:16]	Reserved	RAZ
[15:0]	INSTRLOCK111	Use when **AyUSERSx[7:5]** = `0b111`

Note

- If the `pl310_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 `pl310_LOCKDOWN_BY_MASTER` is not enabled. This corresponds to the simple Lockdown by Way, see *Lockdown by way* on page 2-18.

Replacement strategy

Bit [25] of the Auxiliary Control Register configures the replacement strategy. It can be either round-robin or pseudo-random using an lfsr. The round-robin replacement strategy fills invalid and unlocked ways first; for each line, when ways are all valid or locked, the victim is chosen as the next unlocked way. The pseudo-random replacement strategy fills invalid and unlocked ways first; for each line, when ways are all valid or locked, the victim is chosen randomly between unlocked ways.

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 `pl310_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.

Note

Because the input pins provide the reset values of the address filtering registers, it is not expected that the values of these registers are changed dynamically after reset. Furthermore, changing these values without special attention can lead to unpredictable behavior in some systems.

Address Filtering Start Register

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

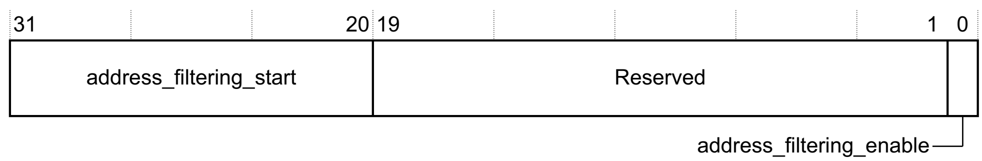


Figure 3-13 Address Filtering Start Register bit assignments

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 for bits [31:20] of the filtering 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

ARM recommends 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 3-14 on page 3-33 shows the register bit assignments.

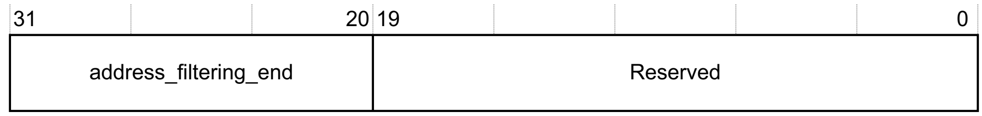


Figure 3-14 Address Filtering End Register bit assignments

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 for bits [31:20] of the filtering 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 and 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 3-15 shows the register bit assignments.

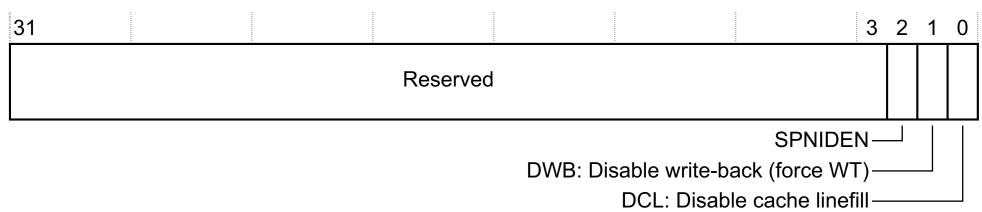


Figure 3-15 Debug Control Register bit assignments

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 cacheability properties, such as Force Write-Allocate and exclusive cache configuration.

3.3.14 Prefetch Control Register

The Prefetch Control Register characteristics are:

Purpose Enables prefetch-related features that can improve system performance.

Usage constraints This register has both read-only, non-secure, and read and write, secure, permissions. Any secure or non-secure access can read this register. Only a secure access can write to this register. If a non-secure access attempts to write to this register, the register issues a DECERR response and does not update.

Note

You must preserve the reserved bits when you write to this register.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-4.

Figure 3-16 on page 3-35 shows the Prefetch Control Register bit assignments.

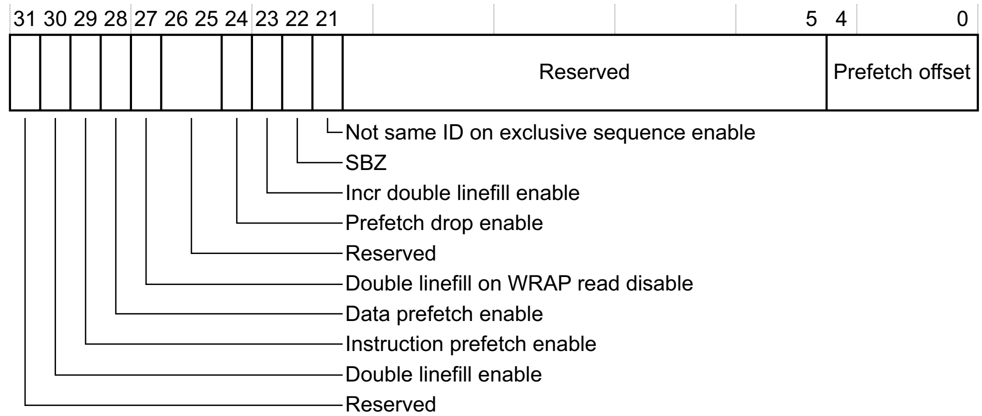


Figure 3-16 Prefetch Control Register bit assignments

Table 3-39 shows the register bit assignments.

Table 3-39 Prefetch Control Register bit assignments

Bits	Field	Description
[31]	Reserved	SBZ/RAZ
[30]	Double linefill enable	You can set the following options for this register bit: `0` The L2CC always issues 4x64-bit read bursts to L3 on reads that miss in the L2 cache. This is the default. `1` The L2CC issues 8x64-bit read bursts to L3 on reads that miss in the L2 cache.
[29]	Instruction prefetch enable[a]	You can set the following options for this register bit: `0` Instruction prefetching disabled. This is the default. `1` Instruction prefetching enabled.
[28]	Data prefetch enable[a]	You can set the following options for this register bit: `0` Data prefetching disabled. This is the default. `1` Data prefetching enabled.
[27]	Double linefill on WRAP read disable	You can set the following options for this register bit: `0` Double linefill on WRAP read enabled. This is the default. `1` Double linefill on WRAP read disabled.
[26]	Reserved	SBZ/RAZ if you define the `pl310_SPECULATIVE_READ` synthesis option. SBO/RAO if you do not define the `pl310_SPECULATIVE_READ` synthesis option.
[25]	Reserved	SBZ/RAZ
[24]	Prefetch drop enable	You can set the following options for this register bit: `0` The L2CC does not discard prefetch reads issued to L3. This is the default. `1` The L2CC discards prefetch reads issued to L3 when there is a resource conflict with explicit reads.

Table 3-39 Prefetch Control Register bit assignments (continued)

Bits	Field	Description
[23]	Incr double Linefill enable	You can set the following options for this register bit: `0` The L2CC does not issue INCR 8x64-bit read bursts to L3 on reads that miss in the L2 cache. This is the default. `1` The L2CC can issue INCR 8x64-bit read bursts to L3 on reads that miss in the L2 cache.
[22]	Reserved	SBZ/RAZ
[21]	Not same ID on exclusive sequence enable	You can set the following options for this register bit: `0` Read and write portions of a non-cacheable exclusive sequence have the same AXI ID when issued to L3. This is the default. `1` Read and write portions of a non-cacheable exclusive sequence do not have the same AXI ID when issued to L3.
[20:5]	Reserved	SBZ/RAZ
[4:0]	Prefetch offset	Default value = `0b00000`. **Note** You must only use the Prefetch offset values of 0-7, 15, 23, and 31 for these bits. The L2C-310 does not support the other values.

a. You can access these bits by using both the Auxiliary Control Register, see *Auxiliary Control Register* on page 3-10, and the Prefetch Control Register. You cannot modify the Auxiliary Control register when the L2 cache is enabled. You can modify the Prefetch Control Register in all conditions.

See *AXI locked and exclusive accesses* on page 2-9 for more information.

3.3.15 Power Control Register

The pwr_ctrl Register characteristics are:

Purpose Controls the operating mode clock and power modes.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 3-2 on page 3-4.

Figure 3-17 shows the pwr_ctrl Register bit assignments.

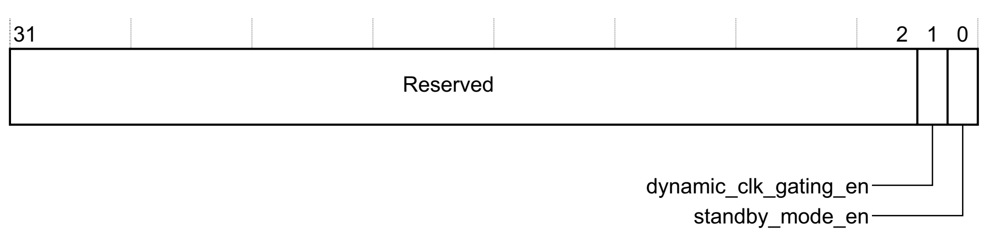


Figure 3-17 pwr_ctrl Register bit assignments

Table 3-40 shows the pwr_ctrl Register bit assignments.

Table 3-40 pwr_ctrl Register bit assignments

Bits	Field	Description	
[31:2]	Reserved	SBZ/RAZ	
[1]	dynamic_clk_gating_en	Dynamic clock gating enable.	
		`0`	Disabled. This is the default.
		`1`	Enabled.
[0]	standby mode en	Standby mode enable.	
		`0`	Disabled. This is the default.
		`1`	Enabled.

See *Power modes* on page 2-47 for more information.

Appendix A
Signal Descriptions

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

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
CLKSTOPPED	Output	Indicates L2C-310 clock is stopped
DATACLKEN	Input	Clock enable for Data RAM interface
DATACLKOUT[a]	Output	Clock for Data RAM
DATACLKOUT[3:0][b]		
DATACLKOUTEN[a]	Output	Clock enable for Data RAM clock
DATACLKOUTEN[3:0][b]		
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
STOPCLK	Input	Request to stop L2C-310 clock
TAGCLKEN	Input	Clock enable for tag RAM interface
TAGCLKOUT	Output	Clock for tag RAM
TAGCLKOUTEN	Output	Clock enable for tag RAM clock

a. Without banking.
b. With banking.

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 *Auxiliary Control Register* on page 3-10.
CACHEID[5:0]	Input	Cache controller cache ID.
CFGADDRFILTEN[a]	Input	Address filtering Enable out of reset.
CFGADDRFILTEND[11:0][a]	Input	Address filtering End Address out of reset.
CFGADDRFILTSTART[11:0][a]	Input	Address filtering Start Address out of reset.
CFGBIGEND	Input	Set HIGH to enable big-endian mode for accessing configuration registers out of reset. Set LOW to enable little-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.
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.
WAYSIZE[2:0]	Input	Size of ways for Auxiliary Control Register. See *Auxiliary Control Register* on page 3-10.

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-5
- *Master port 0* on page A-6
- *Master port 1* on page A-7.

Note

Signals **AWSIZESx[2]**, **WIDx[5:0]**, and **ARSIZESx[2]**,where x = 0 or 1, are not used internally.

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[`pl310_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][a]	Input	Hint signals from ARM Cortex-A9 processor
ARUSERS0[7:5][a]	Input	Master ID for lockdown by master
ARUSERS0[4:1][a]	Input	Inner cacheable attributes
ARUSERS0[0][a]	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
AWIDS0[`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

Table A-3 Slave port 0 signals (continued)

Signal	Type	Description
AWSIZES0[2:0]	Input	Burst size
AWUSERS0[11:10][a]	Input	Hint signals from ARM Cortex-A9 processor
AWUSERS0[9][a]	Input	Indicates a clean eviction[b]
AWUSERS0[8][a]	Input	Indicates an eviction[b]
AWUSERS0[7:5][a]	Input	Master ID for lockdown by master
AWUSERS0[4:1][a]	Input	Inner cacheable attributes
AWUSERS0[0][a]	Input	Shared attribute
AWVALIDS0	Input	Address valid
BIDS0[`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[`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
SRENDS0[3:0]	Output	Output speculative read ending
SRIDS0[`pl310_4SR_ID_MAX:0][c]	Output	Output speculative read ID
WDATAS0[63:0]	Input	Write data bus
WIDS0[`pl310_AXI_ID_MAX:0]	Input	Write ID
WLASTS0	Input	Write last transfer
WREADYS0	Output	Write data accepted
WSTRBS0[7:0]	Input	Write strobes
WVALIDS0	Input	Write data valid

a. Take care when you connect these USER pins to the A9 pins because they are not one-to-one connections.

b. Exclusive cache configuration only.

c. PL310_4SR_ID_MAX=4 x (`pl310_AXI_ID_MAX+1)-1, as `pl310_defs.v` specifies.

A.3.2 Slave port 1

Slave port 1 is only implemented in a two-slave configuration. Slave port 1 signals are the same as slave port 0 signals except that **S0** in the signal names are replaced with **S1**.

A.3.3 Master port 0

Table A-4 shows the master port 0 signals.

Table A-4 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
ARSIZEM0[2:0]	Output	Burst size
ARUSERM0[a]	Output	ID indication of L1 originating transaction
ARVALIDM0	Output	Address valid
AWADDRM0[31:0]	Output	Address bus
AWBURSTM0[1:0]	Output	Burst type
AWCACHEM0[3:0]	Output	Cache information
AWIDM0[`pl310_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
AWUSERM0[a]	Output	ID indication of L1 originating transaction
AWVALIDM0	Output	Address valid
BIDM0[`pl310_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[`pl310_AXI_ID_MAX+2:0]	Input	Read ID
RLASTM0	Input	Read last transfer
RREADYM0	Output	Read accepted

Table A-4 Master port 0 signals (continued)

Signal	Type	Description
RRESPM0[1:0]	Input	Read response
RVALIDM0	Input	Read data valid
WDATAM0[63:0]	Output	Write data bus
WIDM0[`pl310_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. Implemented if you define the pl310_ID_ON_MASTER_IF synthesis option.

A.3.4 Master port 1

Master port 1 is only implemented in a two-master configuration. Master port 1 signals are the same as master port 0 signals except that **M0** in the signal names are replaced with **M1**.

A.4 RAM interface

The RAM interface consists of the following sub interfaces:

- *Data RAM interface*
- *Tag RAM interface* on page A-9.

A.4.1 Data RAM interface

Table A-5 shows the data RAM interface signals.

Table A-5 Data RAM interface signals

Signal	Type	Description
DATAADDR[17:0][a] **DATAADDR[16:0]**[b] **DATAADDR[15:0]**[c] **DATAADDR[14:0]**[d]	Output	Data RAM address
DATACS[e] **DATACS[3:0]**[f]	Output	Data RAM chip select
DATAEN[31:0]	Output	Data RAM byte write enables
DATAERR[e] **DATAERR[3:0]**[f]	Input	Data RAM error
DATAnRW	Output	Data RAM write control signal
DATAPRD[31:0][g]	Input	Data RAM parity read data
DATAPWD[31:0][g]	Output	Data RAM parity write data
DATARD[255:0]	Input	Data RAM read data
DATARDBANKSEL[3:0][f]	Output	Data RAM bank select
DATAWAIT	Input	Data RAM wait
DATAWD[255:0]	Output	Data RAM write data

a. For a 16-way implementation, without banking.
b. For an 8-way implementation, without banking.
c. For a 16-way implementation, with banking.
d. For an 8-way implementation, with banking.
e. Without banking.
f. With banking.
g. Optional. Only present if `pl310_PARITY` is defined.

A.4.2 Tag RAM interface

Table A-6 shows the tag RAM interface signals.

Table A-6 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
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

Table A-6 Tag RAM interface (continued)

Signal	Type	Description
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 a 16-way implementation.

b. For an 8-way implementation.

c. Optional. Only present if `pl310_LOCKDOWN_BY_LINE` is defined.

d. Optional. Only present if `pl310_PARITY` is defined.

e. Only present for a 16-way implementation.

A.5 Cache event monitoring

Table A-7 shows the cache event monitoring signals. See also *Cache event monitoring* on page 2-42.

Table A-7 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
EPFALLOC	Output	Prefetch hint allocated into the L2 cache
EPFHIT	Output	Prefetch hint hits in the L2 cache
EPFRCVDS0	Output	Prefetch hint received by slave port S0
EPFRCVDS1	Output	Prefetch hint received by slave port S1
IPFALLOC	Output	Allocation of a prefetch generated by L2C-310 into the L2 cache
IRHIT	Output	Instruction read hit
IRREQ	Output	Instruction read request
SPNIDEN	Input	Secure privileged non-invasive debug enable
SRCONFS0	Output	Speculative read confirmed in slave port S0
SRCONFS1	Output	Speculative read confirmed in slave port S1
SRRCVDS0	Output	Speculative read received by slave port S0
SRRCVDS1	Output	Speculative read received by slave port S1
WA	Output	Write allocate

A.6 Cache interrupt

Table A-8 shows the cache interrupt signals.

Table A-8 Cache interrupt signals

Signal	Type	Description
DECERRINTR	Output	Decode error received on master port from L3
ECNTRINTR	Output	Event Counter Overflow or Event Counter 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-9 shows the MBIST interface signals.

Table A-9 MBIST interface signals

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

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	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%
CLKSTOPPED	Output	70%
DATAREADLAT[2:0]	Output	70%
DATASETUPLAT[2:0]	Output	70%
DATAWRITELAT[2:0]	Output	70%
IDLE	Output	70%
nRESETn	Input	20%
REGFILEBASE	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 input and output signal timing parameters

Table B-2 shows the slave port 0 input and output signal timing parameters.

Table B-2 Slave port 0 inputs and outputs

Port name	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%
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%

Table B-2 Slave port 0 inputs and outputs (continued)

Port name	Type	Maximum constraint
RLASTS0	Output	70%
RREADYS0	Input	50%
RRESPS0[1:0]	Output	50%
RVALIDS0	Output	70%
SRENDS0[3:0]	Output	70%
SRIDS0[23:0]	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 input and output signal timing parameters

Slave port 1 signals are the same as slave port 0 signals except that **S0** in the signal names are replaced with **S1**.

B.4 Master port 0 input and output signal timing parameters

Table B-3 shows the master port 0 input and output signal timing parameters.

Table B-3 Master port 0 inputs and outputs

Port name	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%
ARUSERM0[7:0]	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%
AWUSERM0[7:0]	Output	70%
AWVALIDM0	Output	70%
BIDM0[7:0]	Input	50%
BREADYM0	Output	70%
BRESPM0[1:0]	Input	50%
BVALIDM0	Input	50%
INCLKENM0	Input	30%
OUTCLKENM0	Output	30%
RDATAM0[63:0]	Input	50%
RIDM0[7:0]	Input	50%

Table B-3 Master port 0 inputs and outputs (continued)

Port name	Type	Maximum constraint
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 input and output signal timing parameters

Master port 1 signals are the same as master port 0 signals except that **M0** in the signal names are replaced with **M1**.

B.6 RAMs signal timing parameters

This section shows the RAMs signal timing parameters in:

- *Data RAM input and output signal timing parameters*
- *Tag RAM input and output signal timing parameters* on page B-10.

B.6.1 Data RAM input and output signal timing parameters

Table B-4 shows the Data RAM input and output signal timing parameters.

Table B-4 Data RAM inputs and outputs

Port name	Type	Maximum constraint
DATAADDR[17:0][a]	Output	70%
DATAADDR[16:0][b]		
DATAADDR[15:0][c]		
DATAADDR[14:0][d]		
DATACLKEN	Input	30%
DATACLKOUT	Output	50%
DATACLKOUTEN	Output	50%
DATACS[e]	Output	70%
DATAEN[31:0]	Output	70%
DATAERR[e]	Input	50%
DATAnRW	Output	70%
DATAPEN[31:0]	Output	70%
DATAPnRW	Output	70%
DATAPRD[31:0][f]	Input	50%
DATAPWD[31:0][f]	Output	70%
DATARD[255:0]	Input	50%
DATARDBANKSEL[3:0][g]	Output	70%
DATAWAIT	Input	30%
DATAWD[255:0]	Output	70%

a. For a 16-way implementation, without banking.
b. For an 8-way implementation, without banking.
c. For a 16-way implementation, with banking.
d. For an 8-way implementation, with banking.
e. Without banking.
f. Optional. Only present if `pl310_PARITY` is defined.
g. With banking.

B.6.2 Tag RAM input and output signal timing parameters

Table B-5 shows the Tag RAM input and output signal timing parameters.

Table B-5 Tag RAM inputs and outputs

Port name	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%
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 input and output signal timing parameters

Table B-6 shows the event monitor input and output signal timing parameters.

Table B-6 Event monitor inputs and outputs

Port name	Type	Maximum constraint
CO	Output	70%
DRHIT	Output	70%
DRREQ	Output	70%
DWHIT	Output	70%
DWREQ	Output	70%
DWTREQ	Output	70%
EPFALLOC	Output	70%
EPFHIT	Output	70%
EPFRCVDS0	Output	70%
EPFRCVDS1	Output	70%
IPFALLOC	Output	70%
IRHIT	Output	70%
IRREQ	Output	70%
SRCONFS0	Output	70%
SRCONFS1	Output	70%
SRRCVDS0	Output	70%
SRRCVDS1	Output	70%
SPNIDEN	Input	70%
WA	Output	70%

B.8 Cache interrupt ports signal timing parameters

Table B-7 shows the cache interrupt ports signal timing parameters.

Table B-7 Cache interrupt ports

Port name	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 input and output signal timing parameters

Table B-8 shows the MBIST interface input and output signal timing parameters.

Table B-8 MBIST interface signal inputs and outputs

Port name	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%
MTESTON	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:

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.

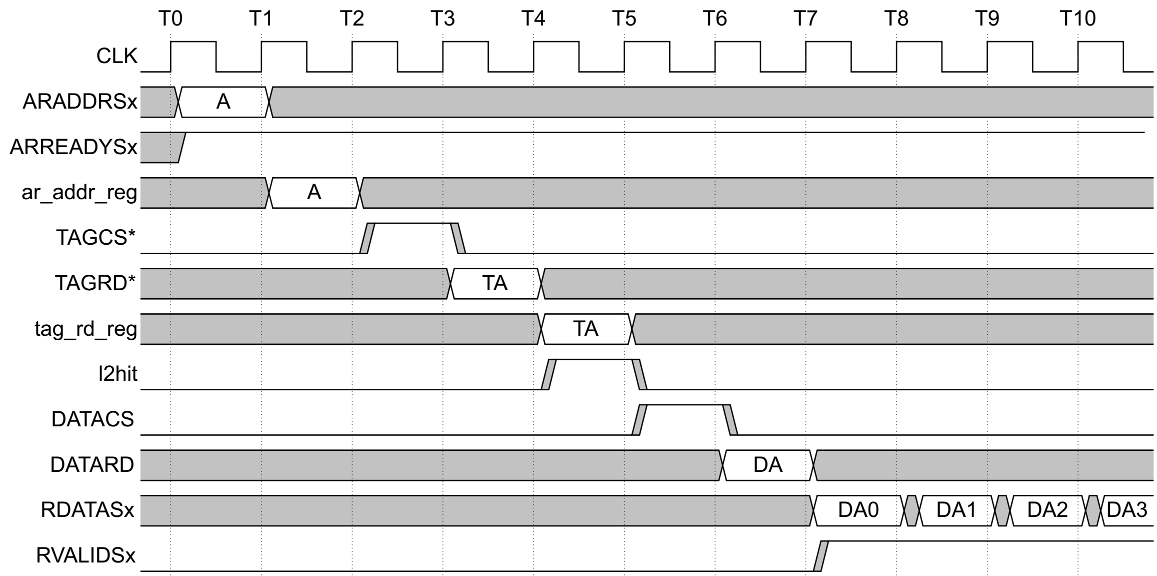


Figure C-1 Single read hit transaction

C.2 Single read miss transaction

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

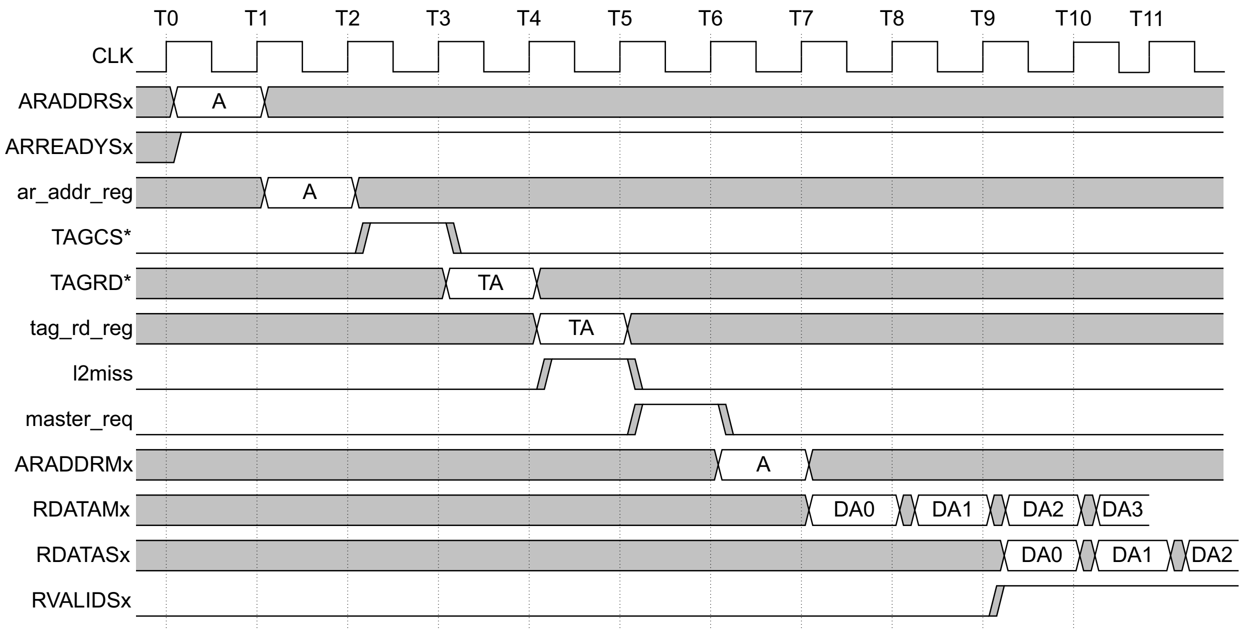


Figure C-2 Single read miss transaction

C.3 Single non-cacheable read transaction

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

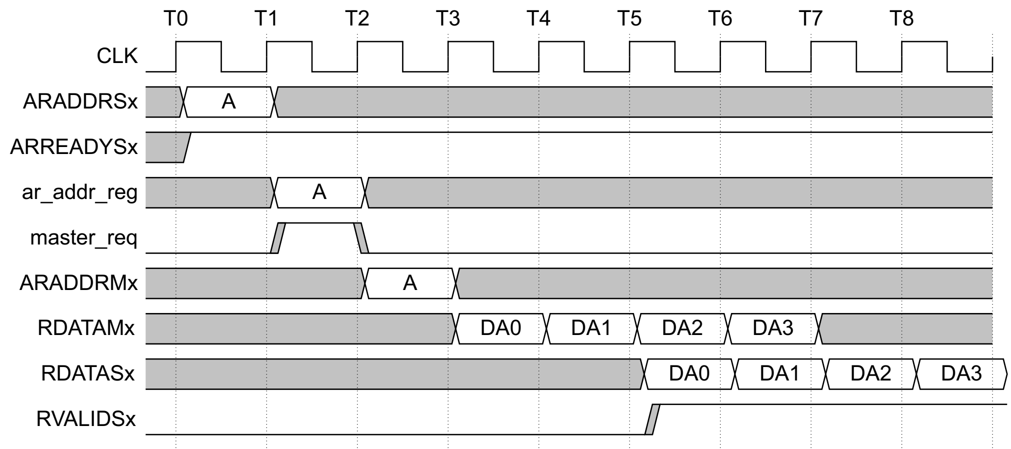


Figure C-3 Single non-cacheable read transaction

C.4 Outstanding read hit transactions

Figure C-4 and Figure C-5 show the timing for an outstanding read hit transaction.

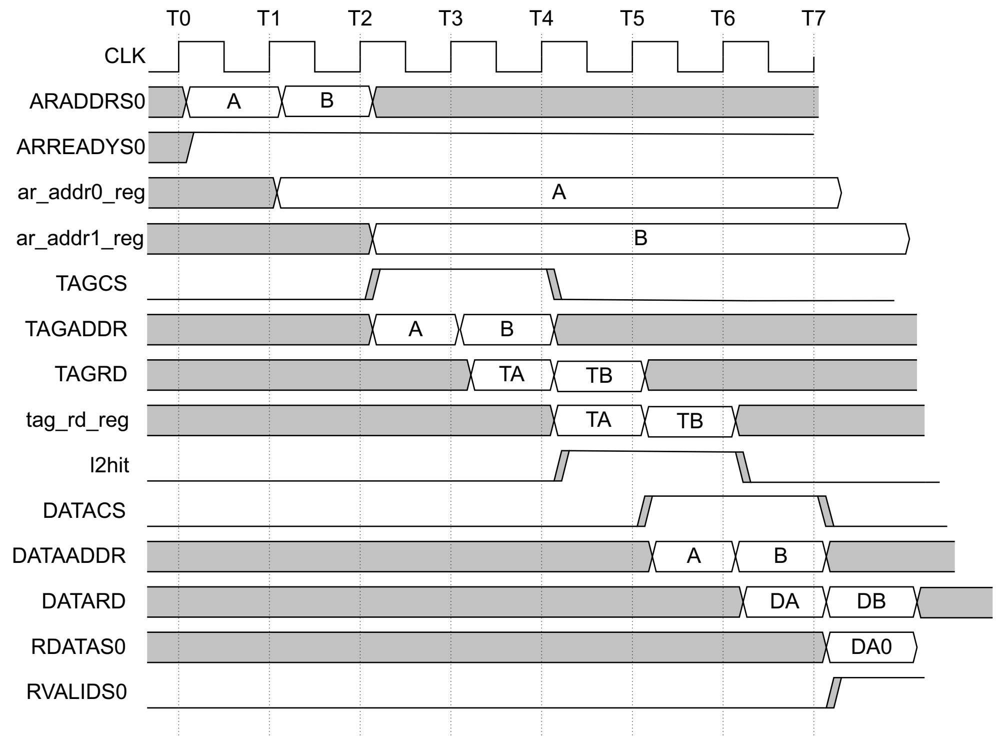


Figure C-4 Outstanding read hit transactions, part 1

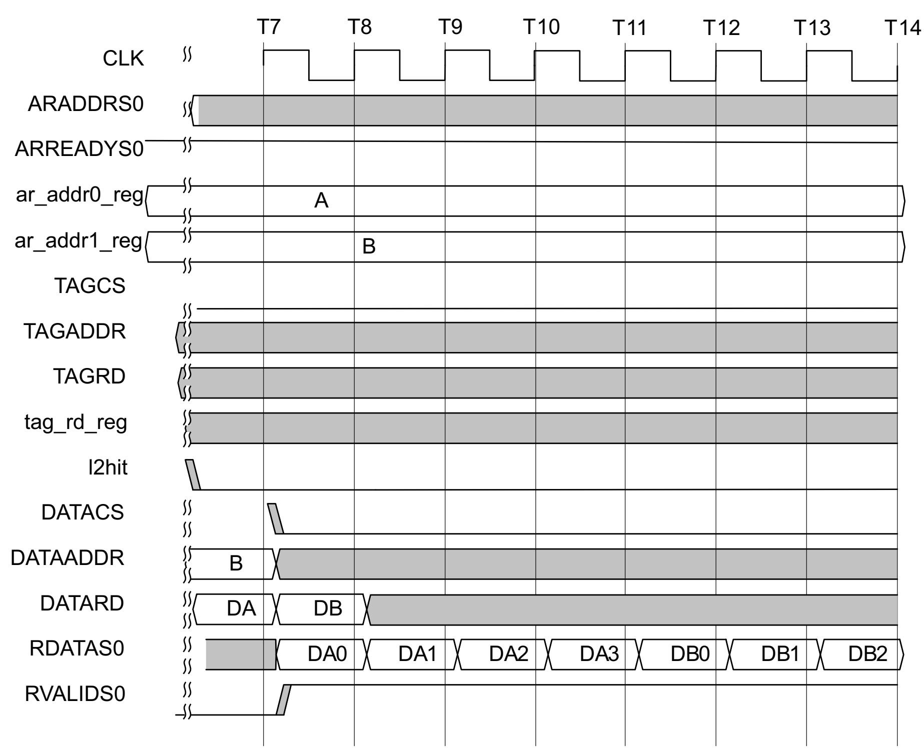


Figure C-5 Outstanding read hit transactions, part 2

C.5 Hit under miss read transactions

Figure C-6 and Figure C-7 on page C-7 shows the timing for a hit under miss read transaction case.

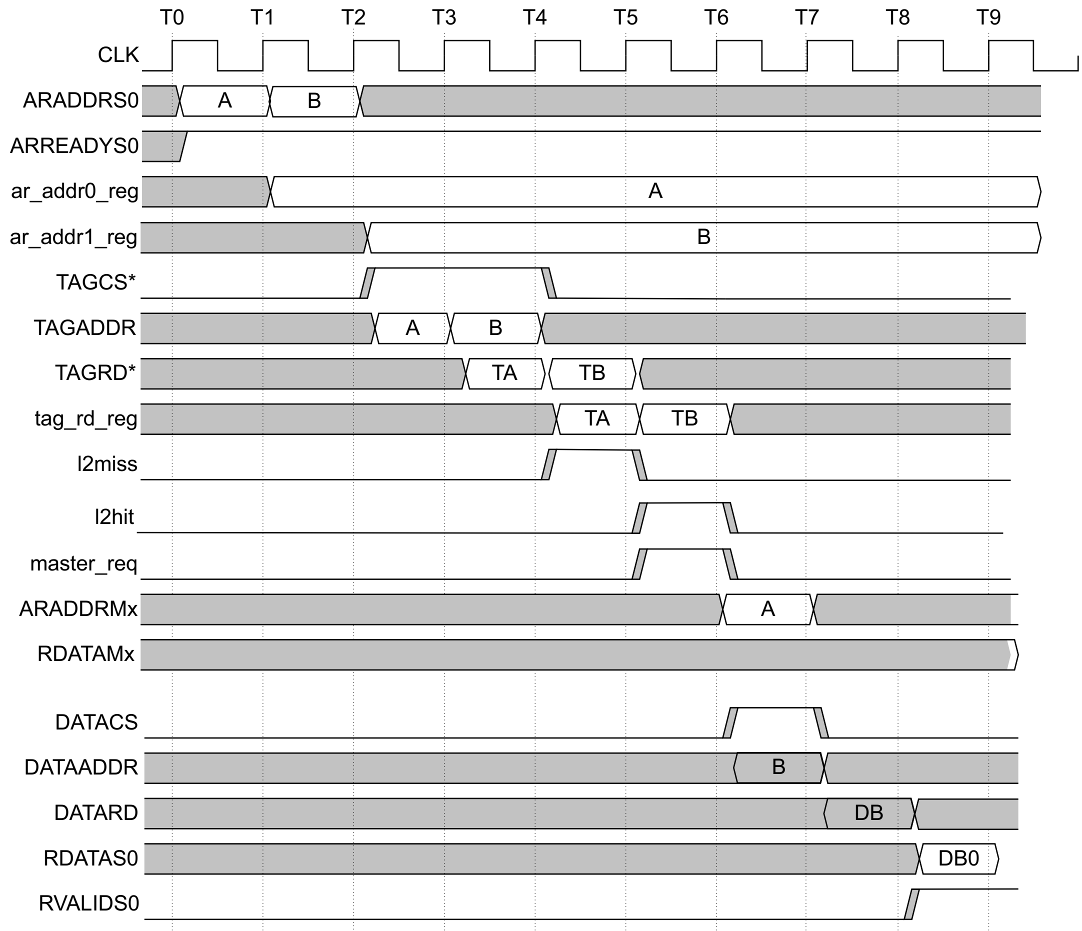


Figure C-6 Hit under miss read transactions, part 1

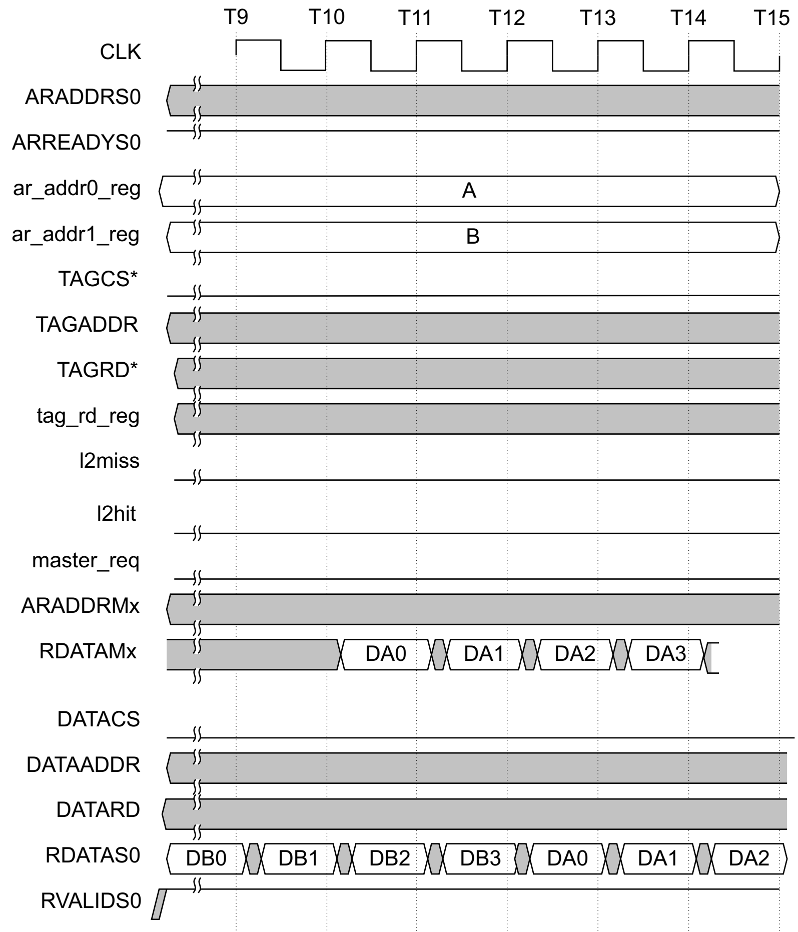


Figure C-7 Hit under miss read transactions, part 2

C.6 Single bufferable write transaction

Figure C-8 shows the timing for a single bufferable write transaction.

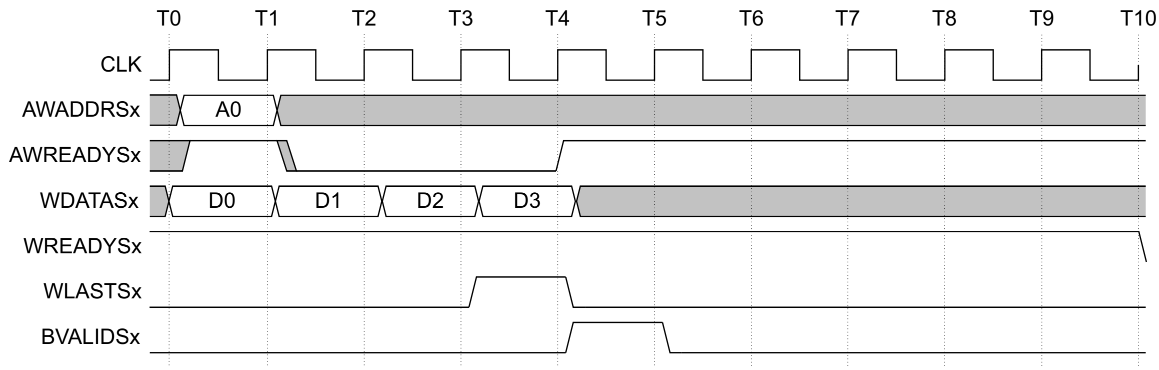


Figure C-8 Single bufferable write transaction

C.7 Single non-bufferable write transaction

Figure C-9 shows the timing for a single non-bufferable write transaction.

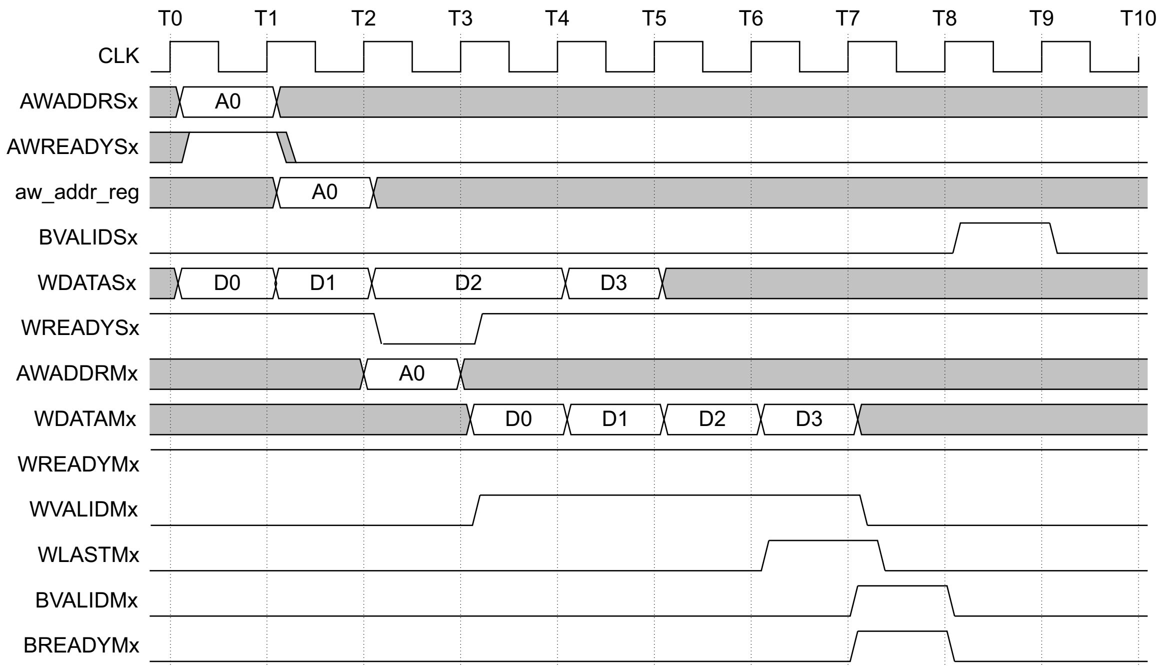


Figure C-9 Single non-bufferable write transaction

Appendix D
Revisions

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

Table D-1 Differences between issue C and issue D

Change	Location	Affects
Cache features updated	*Cache configurability* on page 2-2.	r3p0
Note added	*AXI master and slave interfaces* on page 2-3.	All revisions
Write acceptance capability updated	Table 2-3 on page 2-4.	All revisions
Exclusive cache configuration clarified	*Exclusive cache configuration* on page 2-17.	All revisions
RAM organization updated	*RAM organization* on page 2-22	r3p0
Figure updated	Figure 2-13 on page 2-27	r3p0
Cortex-A9 optimizations updated	*Cortex-A9 optimizations* on page 2-37	r3p0
Event pins added	Table 2-21 on page 2-42	r3p0
Event pin deleted	Table 2-21 on page 2-42	r3p0
Interrupt pin ERRWTINTR description updated	Table 2-22 on page 2-43	r3p0
Section added	*Dynamic clock gating* on page 2-47	r3p0
Section updated	*Standby mode* on page 2-47	r3p0
Example cache controller start-up programming sequence updated	*Initialization sequence* on page 3-2	r3p0
Register map updated	Table 3-1 on page 3-4	r3p0

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

Change	Location	Affects
Register summary updated	Table 3-2 on page 3-4	r3p0
Register bit assignments updated	Table 3-3 on page 3-7	r3p0
Register updated	*Cache Type Register* on page 3-7	r3p0
Register updated	*Auxiliary Control Register* on page 3-10	r3p0
reg2_ev_counter0_cfg and reg2_ev_counter1_cfg Register bit assignments updated	Table 3-9 on page 3-15	r3p0
Cache Maintenance Operations table updated	Table 3-15 on page 3-22	r3p0
Section clarified	*Invalidate Operations* on page 3-25	All revisions
Clock and reset signals updated	Table A-1 on page A-2	r3p0
Slave port 0 signals table updated	Table A-3 on page A-4	r3p0

Table D-2 Differences between issue D and issue E

Change	Location	Affects
Removal of AXI ID encoding tables	*Master and slave port IDs* on page 2-7	r3p1
Description of a new bit in the Prefetch Control Register	*Prefetch Control Register* on page 3-34	r3p1

Table D-3 Differences between issue E and issue F

Change	Location	Affects
Hazard checking description clarified	*Hazards* on page 2-36	r3p1 and r3p2
MBISTDCTL interface signals updated	Table A-9 on page A-13	r3p1 and r3p2
Preset offset bit description clarified	Table 3-39 on page 3-35	r3p1 and r3p2
CFGBIGEND signal description clarified	Table A-2 on page A-3	r3p1 and r3p2
Bit 25 of reg1_aux_control Register bit assignments Register updated	Figure 3-4 on page 3-10	r3p1 and r3p2
Speculative read feature clarified	*Cortex-A9 optimizations* on page 2-37	r3p1 and r3p2
Reset procedure added	*Reset requirement* on page 2-35	r3p2

Table D-4 Differences between issue F and issue G

Change	Location	Affects
Additional bullet added to note in Double Linefill issuing section. The bullet explains that Double linefills only occur if a WRAP4 or an INCR4 64-bit transaction is received on the slave ports.	*Double linefill issuing* on page 2-39	r3p0 to r3p2
Data ram interface behavior clarified.	*RAM latencies* on page 2-32	r3p0 to r3p2
Section added.	*Tag lock RAM* on page 2-27	r3p0 to r3p2
Section clarified.	*AXI exclusive accesses* on page 2-10	r3p0 to r3p2

Table D-4 Differences between issue F and issue G (continued)

Change	Location	Affects
DATARDBANKSEL signal description clarified.	Figure 2-8 on page 2-22 Figure 2-10 on page 2-25 Table A-5 on page A-8 Table B-4 on page B-9	r3p0 to r3p2
Master port behavior table updated.	Table 2-19 on page 2-39	r3p0 to r3p2
Interface signals list updated.	Table A-9 on page A-13	r3p0 to r3p2
Section updated.	*Store buffer operation* on page 2-35	r3p0 to r3p2

Table D-5 Differences between issue G and issue H

Change	Location	Affects
Combined acceptance capability attribute clarified	Table 2-3 on page 2-4	r3p0 to r3p3
Force write allocate section updated	*Force write allocate* on page 2-16	r3p0 to r3p3
Double linefill issuing behavior updated.	*Double linefill issuing* on page 2-39	r3p0 to r3p3
Parity and RAM error support behavior updated	*Parity and RAM error support* on page 2-44 Table 2-23 on page 2-45 Table 2-24 on page 2-45 Table 2-25 on page 2-45 Table 2-26 on page 2-46 Table 2-28 on page 2-46	r3p0 to r3p3
Cache maintenance operations behavior clarified	*Background operations* on page 3-23	r3p0 to r3p3
Reset sequence updated	*Reset requirement* on page 2-35	r3p0 to r3p3