L220 Cache Controller

Revision: r1p7

Technical Reference Manual

ARM®

ARM DDI 0329L

L220 Cache Controller
Technical Reference Manual

Release Information

Change history

Date	Issue	Confidentiality	Change
09 August 2004	A	Non-Confidential	First release.
12 November 2004	B	Non-Confidential	Update for r1p0. No change in description of functionality.
10 December 2004	C	Non-Confidential	Update for r1p1. No change in description of functionality.
28 January 2005	D	Non-Confidential	Update for r1p2. Added description for bit [12] in *Register 1, Auxiliary Control Register* on page 3-10.
02 August 2005	E	Non-Confidential	Update for r1p3. No change in description of functionality.
19 December 2005	F	Non-Confidential	Update for r1p4. No change in description of functionality. "Exclusive cache operation" changed to "exclusive cache configuration" throughout. Timing diagram appendix added.
01 March 2006	G	Non-Confidential	Additions to error response table and clarification of use of C bit in Register 7. Part number corrected, timing diagrams updated. Security Override Check feature redefined. **L220CLAMP** added to Table A-2 on page A-3.
04 May 2006	H	Non-Confidential	Update for r1p5. Peripheral Port behavior redefined. Table 2-1 on page 2-9 and Table 2-2 on page 2-10, AXI attributes tables, added. *Exclusive cache configuration* on page 2-15 rewritten for clarity. Timing diagram *Single bufferable write transaction* on page C-6 updated
08 September 2006	I	Non-Confidential	Update for r1p7. Clock enable usage model changed.
22 February 2007	J	Non-Confidential	Updated to correct defects. No change in description of functionality.
30 October 2007	K	Non-Confidential	Correction to Cache Synchronization description in *Background line operations* on page 3-23.
14 December 2007	L	Non-Confidential	Amend Cache Synchronization description in *Background line operations* on page 3-23.

Proprietary Notice

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

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

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

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Contents
L220 Cache Controller Technical Reference Manual

List of Tables
L220 Cache Controller Technical Reference Manual

List of Figures
L220 Cache Controller Technical Reference Manual

Preface

This preface introduces the *L220 Cache Controller Technical Reference Manual*. It contains the following sections:

- *About this manual* on page xiv
- *Feedback* on page xviii.

About this manual

This is the *Technical Reference Manual* (TRM) for the *L220 Cache Controller*. In this manual the generic term cache controller means the L220 Cache Controller.

Product revision status

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

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

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

Intended audience

This manual has been written for hardware and software engineers implementing the L220 Level Two Cache Controller into ASIC designs. It provides information to enable designers to integrate the device into a target system as quickly as possible.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the cache controller.

Chapter 2 *Functional Overview*

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

Chapter 3 *Programmer's Model*

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

Appendix A *Signal Descriptions*

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

Appendix B *AC Parameters*

Read this appendix for a description of the AC signal timing parameters

Appendix C *Timing Diagrams*

Read this appendix for a description of cache controller timing diagrams.

Glossary

Read this chapter for a glossary of terms.

Conventions

Conventions that this manual can use are described in:

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

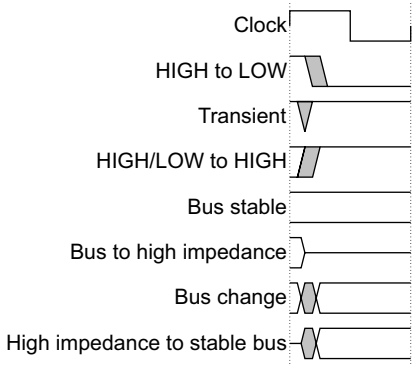
Typographical

The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects which register is accessed.

Timing diagrams

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



key to timing diagram conventions

Signals

The signal conventions are:

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

Prefix A Denotes *Advanced eXtensible Interface* (AXI) global and address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix n Denotes active-LOW signals.

Prefix R Denotes AXI read channel signals.

Prefix W Denotes AXI write channel signals.

Numbering

The numbering convention is:

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

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

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

Further reading

This section lists publications by ARM and by third parties.

See `http://infocenter.arm.com/help/index.jsp` for access to ARM documentation.

ARM publications

This document contains information that is specific to the Level Two Cache Controller. See the following documents for other relevant information:

- *AMBA AXI Protocol* (ARM IHI 0022)
- *ARM Architecture Reference Manual* (ARM DDI 0406)
- *ARM Architecture Reference Manual, Security Extensions supplement* (ARM DDI 0309)
- *ARM L220 MBIST Technical Reference Manual* (ARM DDI 0330).
- *ARM L220 Cache Controller Implementation Guide* (ARM DII 0104)
- *ARM1176JZF-S Technical Reference Manual* (ARM DDI 0301)
- *ARM1176JZ-S Technical Reference Manual* (ARM DDI 0333)
- *ARM Limited, Level 2 Cache for High-performance ARM Core-based SoC Systems,* January 2004.

Feedback

ARM welcomes feedback on the L220 Cache Controller and its documentation.

Feedback on this product

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

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

Feedback on this manual

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

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

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the L220 Cache Controller and its features. It contains the following sections:

1.1 About the Cache Controller

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

The Cache Controller features:

- TrustZone architecture for enhanced OS security
- Master, slave, and peripheral AXI/AMBA interfaces designed for high performance systems
- *Intelligent Energy Manager* (IEM) support.

The Cache Controller is a unified, physically addressed, physically tagged 8-way cache. You can lock the replacement algorithm on a way basis, enabling the associativity to be reduced from eight-way down to one-way, direct mapped. You must use the locked ways as physically mapped memory. 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.

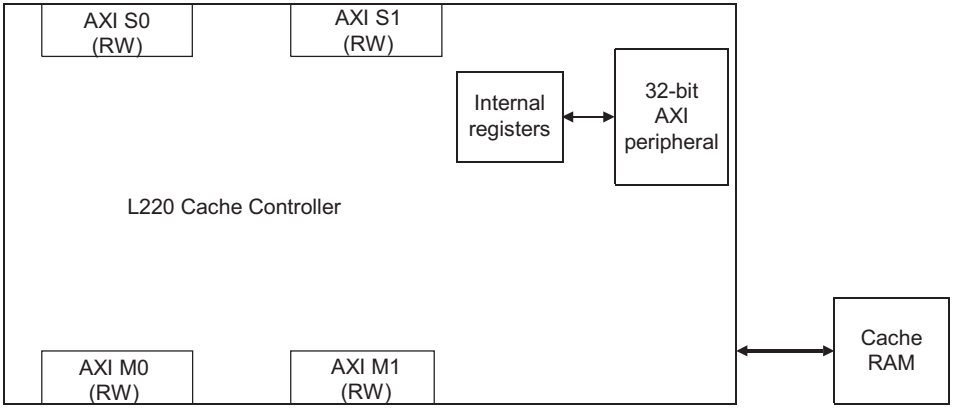


Figure 1-1 Top level diagram

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

Table 1-1 Typical memory sizes and access times

Memory type	Typical size	Typical access time
Processor registers	128KB	1 cycle
On-chip L1 cache	32KB	1-2 cycle(s)
On-chip L2 cache	128KB	8 cycles
Main memory (L3) dynamic RAM	MB or GB[a]	30-42 cycles
Back-up memory (hard disk) (L4)	MB or GB	> 500 cycles

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

1.1.1 Features

The Cache Controller has the following features:

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

  Note

  Lockdown format C supported is also known as way locking.

- L2 cache size can be 16KB to 2MB, depending on the configuration and the use of lockdown registers.
- Direct mapped to 8-way associativity, depending on the use of lockdown registers.
- Fixed line length of 32 bytes (eight words or 256 bits), data RAM is byte writable.
- Supports all of the AXI cache modes:
  - write-through and write-back, through configuration of the **ACACHE** lines.
  - Read allocate, write allocate, read and write allocate.
- Force write allocate option to always have cacheable writes allocated for L2 cache, for cores not supporting this mode.

- Shared transfers treated as noncacheable with the option to override this behavior, also known as Shared Override.
- TrustZone support, with the following features:
  - *Non-Secure* (NS) tag bit added in tag RAM and used for lookup in the same way as an address bit. The NS-tag bit is added in all buffers.
  - NS bit in tag used to determine access to L3 for eviction and WB.
  - Restrictions for NS accesses for control, debug and maintenance registers to restrict access to secure data.
- All read allocate transactions are converted to critical word fill transactions.
- Pseudo-Random victim selection policy. You can make this deterministic with use of lockdown registers.
- Two *Line Fill buffers* (LFBs). These buffers capture linefill data from main memory, waiting for a complete line before writing to L2 memory. This makes the L2 cache non-blocking for requests from the other slave ports.
- Two *Line Read buffers* (LRBs), one in each slave. These buffers hold a line from the L2 cache in case of cache hit.
- One *Eviction buffer* (EB). This holds an evicted line from the L2 cache, to be written back to main memory.
- One *Write buffer* (WB). This holds buffered writes before their draining to main memory, in addition to the L2 cache. The WB is made of 2 slots (256 bits data line and 1 address per slot) and enables multiple writes to a same line to be merged.
- One *Write-Allocate buffer* (WAB). This buffer merges data from the WB and missing data (if WB line is not full) from the M1 master port before requesting an allocation to the cache.
- Option to select one or two master ports.
- Option to select one or two slave ports.

  Note

  If only one slave is supported it is intended that only one master is configured, not two.

- Software option to enable exclusive cache configuration, see *Behavior for ARMv6 transactions* on page 2-12 for more information.
- *Intelligent Energy Manager* (IEM) register slices with support for synchronous and asynchronous interfacing, also with specific support for IEM.

- Ease of configuration for present and future system optimizations.
- Parity and memory error support.
- MBIST support.
- L2 cache event monitoring. Event signals are exported if you want to use these in conjunction with an event monitoring block. Event monitoring is offered in the Cache Controller with two programmable 32-bit counters. Secure event and performance signals are only available when the signal on the **SPNIDEN** pin is configured HIGH.
- Peripheral 32-bit AXI port for register accesses.

The cache controller is highly configurable. You do not have to resynthesize the module to reconfigure the cache controller for other applications. This means that you can harden the cache controller macrocell once, regardless of the number of cache controller configurations you use.

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

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

To date, ARM processors do not support any form of hardware cache coherency maintenance. This manual assumes that all caches have no cache coherency protocol and do no snooping.

The ARM white paper, *Optimizing system architecture for high-performance cores using the ARM Level Two Cache Controller* contains suggestions for obtaining optimal performance from designs incorporating a Cache Controller.

1.2 Typical system configuration

The Cache Controller works efficiently with ARM processors that implement AXI interfaces. It directly interfaces on the data and instruction interface, so that the internal pipelining of the cache controller is optimized to enable the processors to operate at 1-1 clocking. Figure 1-2 shows an example of a Cache Controller interfaced to an ARM processor.

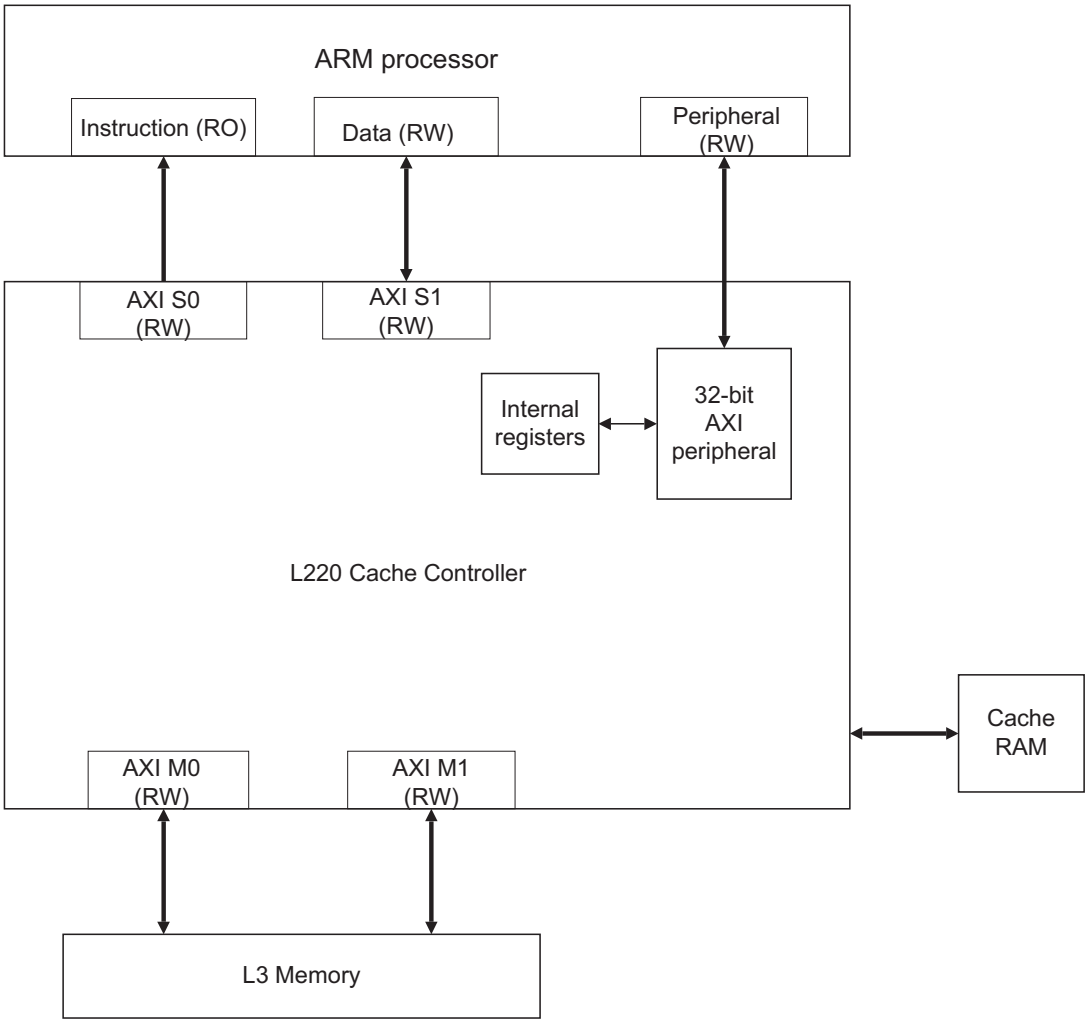


Figure 1-2 Example L220 Cache Controller interfaced to an ARM processor

Note

The slave port **AXI S0** can be used with a RW port, or an additional processor.

Typical processor ports are:

- Instruction port acting as a read-only 64-bit port
- Data port serving as a single read-write 64-bit port
- Peripheral port, that accesses registers that cannot be cached, for example a 32-bit port.

The Cache Controller supports:

- Two read/write 64-bit slave ports for interfacing with Data and Instruction interfaces
- Two read/write 64-bit master ports for interfacing with L3 memory system.

You can configure the Cache Controller to use one or two master ports. Table 1-2 and Table 1-3 show what each master port is used for.

Table 1-2 Master port transactions for a two master port system

Master port 0	Master port 1
Noncached reads and writes from **S0** Linefills with LF buffer 0	Noncached reads and writes from **S1** Linefills with LF buffer 1 Buffered stores with WB or EB Write allocations with LF buffer 1

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

Master port 0	Master port 1
-	Noncached reads and writes from **S1** and **S0** Linefills with LF buffer 1 Buffered stores with WB or EB Write allocations with LF buffer 1

1.3 Peripheral port connectivity with r1

The Cache Controller r1 enables the peripheral port to be optionally connected at L3 in addition to the supported connectivity directly to L1. This configuration enables processors without a dedicated peripheral port to be connected through an interconnect on the master side of the cache controller. Figure 1-3 shows an example of how the peripheral port can be connected. See *Product revisions* on page 1-9 that highlights the changes in the programmer's model between r0 and r1 to support this connectivity within the product.

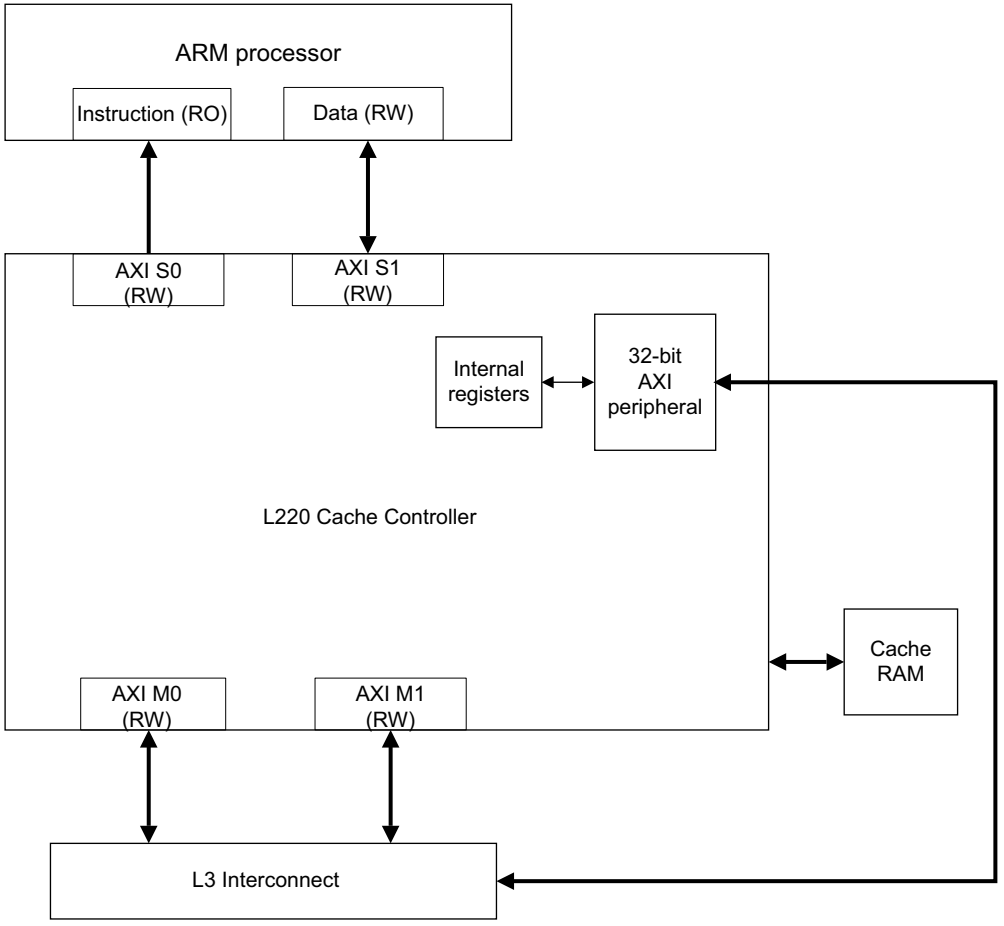


Figure 1-3 Top level view showing ARM processor and peripheral port connectivity

1.4 Product revisions

This is the Technical Reference Manual for the Cache Controller. The differences between product revisions are described in:

- *r0p0 to r1p0*
- *r1p0 to r1p1* on page 1-11
- *r1p1 to r1p2* on page 1-11
- *r1p2 to r1p3* on page 1-11
- *r1p3 to r1p4* on page 1-11
- *r1p4 to r1p5* on page 1-11
- *r1p5 to r1p7* on page 1-12.

1.4.1 r0p0 to r1p0

The differences between these versions are:

- Additional feature point to reference the exclusive cache configuration in *Features* on page 1-3.
- Peripheral port connectivity, see *Peripheral port connectivity with r1* on page 1-8.
- The peripheral port does not support exclusive transfers or locked accesses, see *AXI peripheral port interface* on page 2-4.
- Signals in Figure 2-2 on page 2-5 placed in AXI grouping order.
- A note is added to the *Line read buffer* on page 2-7.
- Addition of an exclusive cache configuration section. See *Behavior for ARMv6 transactions* on page 2-12 and Table 3-6 on page 3-11 for bit 12, exclusive cache configuration.
- A locked access exception is added to *Locked and exclusive accesses* on page 2-17, also the condition relating to the outstanding transactions on the peripheral port is removed.
- Noncacheable transactions are added to *Exported master ports AXI control signals* on page 2-21.
- Minor changes to the section *Configuration and control registers* on page 3-2, including the addition of polling.
- Note changed after Table 3-3 on page 3-7 to reference r1p0 RTL release code.
- Modifications to received transaction sequence in *Register 1,Control Register* on page 3-9.

- To support peripheral port connectivity with r1p0, the L2 Control Register, base offset +0x100, the L220 enable/disable sequence is no longer atomic. You must now poll the register for completion of the cache control sequence. See also *Register 1,Control Register* on page 3-9.
- To support peripheral port connectivity with r1p0, there are significant changes to the behavior of atomic operations. All maintenance operations registers are now read and write. The operations controlled by these registers are now entirely background where the response is given immediately. You must now poll the respective register for completion of the maintenance operation. See also *Register 7, Cache Maintenance Operations* on page 3-22. The following registers no longer control atomic operations:
  - L2 Cache Sync, base offset +0x730.
  - L2 Invalidate Line by PA, base offset +0x770.
  - L2 Clean Line by PA, base offset +0x7B0.
  - L2 Clean Line by Index/Way, base offset +0x7B8.
  - L2 Clean and Invalidate by PA, base offset +0x7F0.
  - L2 Clean and Invalidate Line by Index/Way, base offset +0x7F8.
- To support peripheral port connectivity with r1p0 the following registers no longer have an atomic element:
  - L2 Invalidate by Way, base offset +0x77C.
  - L2 Clean by Way +0x7BC.
  - L2 Clean and Invalidate by Way +0x7FC.

  Each of the operations that these registers control can perform a cache sync but would not return a write response until the operation is completed.

  The operations controlled by these registers are now entirely background where a response is given immediately. You must now poll the respective register for completion of the cache sync element and the background maintenance operation. See also *Register 7, Cache Maintenance Operations* on page 3-22.
- To support peripheral port connectivity with the r1p0, the L2 Test Operation Register, base offset +0xF00, read or write is no longer atomic. You must now poll the register for completion of the read or write test. See also *Register 15, Test and Debug* on page 3-31.
- All **SIDEBAND** signals are now monitored.
- Signal **ARBURSTP[1:0]** is corrected in the signals description.

- The following signals are added:
  - **WRITEBACKS0[1:0]**, see *Behavior for ARMv6 transactions* on page 2-12.
  - **WRITEBACKS1[1:0]**, see *Behavior for ARMv6 transactions* on page 2-12.
  - **RAMCLAMP**, see *Dormant mode considerations* on page 2-29.
- References to C-channel have been changed to AXI low-power interface.

1.4.2 r1p0 to r1p1

There is no change in the described functionality between these versions. Only engineering errata has been incorporated in the product.

1.4.3 r1p1 to r1p2

There is no change in the described functionality between these versions. Only engineering errata has been incorporated in the product.

1.4.4 r1p2 to r1p3

There is no change to the functionality described in this manual. See the engineering errata that accompanies the product deliverables for more information.

1.4.5 r1p3 to r1p4

The functional difference between these versions is that in r1p4 the Security Override Check feature is always enabled and cannot be disabled.

See the release note that accompanies the product deliverables for more information.

See the engineering errata that accompanies the product deliverables for more information about the errata fixes in r1p4.

1.4.6 r1p4 to r1p5

The functional difference between these versions is that in r1p5 the Peripheral Port does not support burst accesses and returns SLVERR when **AxLEN** is different from zero.

The timing diagrams have been updated. See Appendix C *Timing Diagrams*.

See the engineering errata that accompanies the product deliverables for more information about the errata fixes in r1p5.

1.4.7 r1p5 to r1p7

The main functional difference between these versions is that the r1p7 version uses a new clock enable usage model.

See the engineering errata that accompanies the product deliverables for more information about the errata fixes in r1p7.

Chapter 2
Functional Overview

This chapter describes a functional overview and a functional description of the L220 Cache Controller. It contains the following sections:

2.1 Functional description

Figure 2-1 shows the internal data paths of an L220 Cache Controller. It can be used in a system with AXI ARM processors.

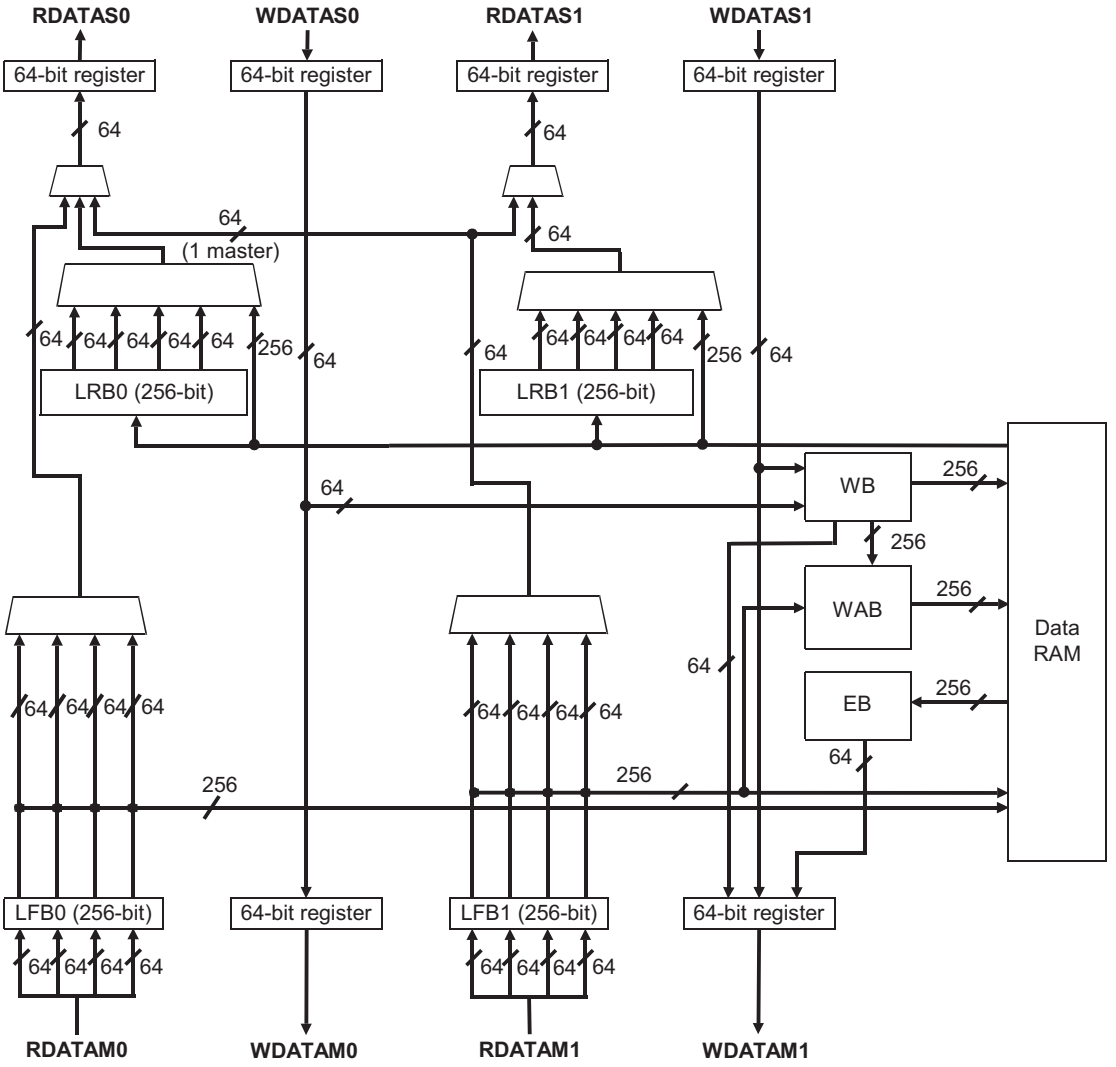


Figure 2-1 Internal data paths

The main functionalities are:

- *Clock enable usage model in the Cache Controller AXI interfaces* on page 2-3

- *Master and slave interfaces*
- *AXI peripheral port interface* on page 2-4
- *TrustZone support* on page 2-5.
- *AXI low-power interface* on page 2-6
- *Internal registers* on page 2-6
- *Buffers* on page 2-6
- *Data RAM* on page 2-8
- *Tag RAM* on page 2-8.

2.1.1 Clock enable usage model in the Cache Controller AXI interfaces

The Cache Controller implements up to five AXI interfaces. Each of these can receive an independent clock enable as follows:

- **ACLKENS0** for the slave 0 (if implemented)
- **ACLKENS1** for the slave 1
- **ACLKENM0** for the master 0 (if implemented)
- **ACLKENM1** for the master 1
- **ACLKENP** for the peripheral port.

These clock enables allow the Cache Controller to communicate with AXI components that run at slower frequencies.

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

- Inputs are sampled on rising edges of **CLK** only when the clock enable is HIGH.
- Outputs are updated on rising edges of **CLK** only when the clock enable is HIGH.

2.1.2 Master and slave interfaces

The masters and slaves are AXI-compatible. The Cache Controller supports:

- Two read-write 64-bit slave ports for interfacing with Data and Instruction interfaces
- Two read-write 64-bit master ports for interfacing with the L3 memory system.

See *AXI master and slave interfaces* on page 2-9 for more information.

2.1.3 AXI peripheral port interface

The AXI peripheral port interface for the Cache Controller is a separate 32-bit AXI slave port. This enables access to the L2 cache control registers specified in Chapter 3 *Programmer's Model*. The AXI peripheral port has the following features:

- 35 × 32-bit read/write registers. A decoder is implemented outside the Cache Controller to produce an **ARVALIDP** or **AWVALDP** signal to select this peripheral.
- The peripheral slave port accepts only a single outstanding address so has a write depth of one.
- 4KB memory footprint.
- Security enabled for writes to the L2 Control and L2 Auxiliary Control registers
- The peripheral slave does not support burst accesses with **AxLENP** different from zero and returns SLVERR response when the length is not zero.
- The peripheral slave does not support exclusive transfers, therefore it responds OKAY.
- The peripheral slave port does not support locked accesses.
- The peripheral slave port does not support accesses with **AxSIZEP** different from 32-bit and returns SLVERR response when the size is not 32-bit.
- The peripheral slave port does not support unaligned accesses and returns a SLVERR response when the address is not word aligned.
- The peripheral slave port does not support write strobes. If a sparse data transfer is made by use of the **WSTRB** signal then the peripheral slave port does not modify the registers and issues a SLVERR write response.
- AXI low power interface.

The interface is shown in Figure 2-2 on page 2-5. See *AXI peripheral port interface operation* on page 2-22 for more information.

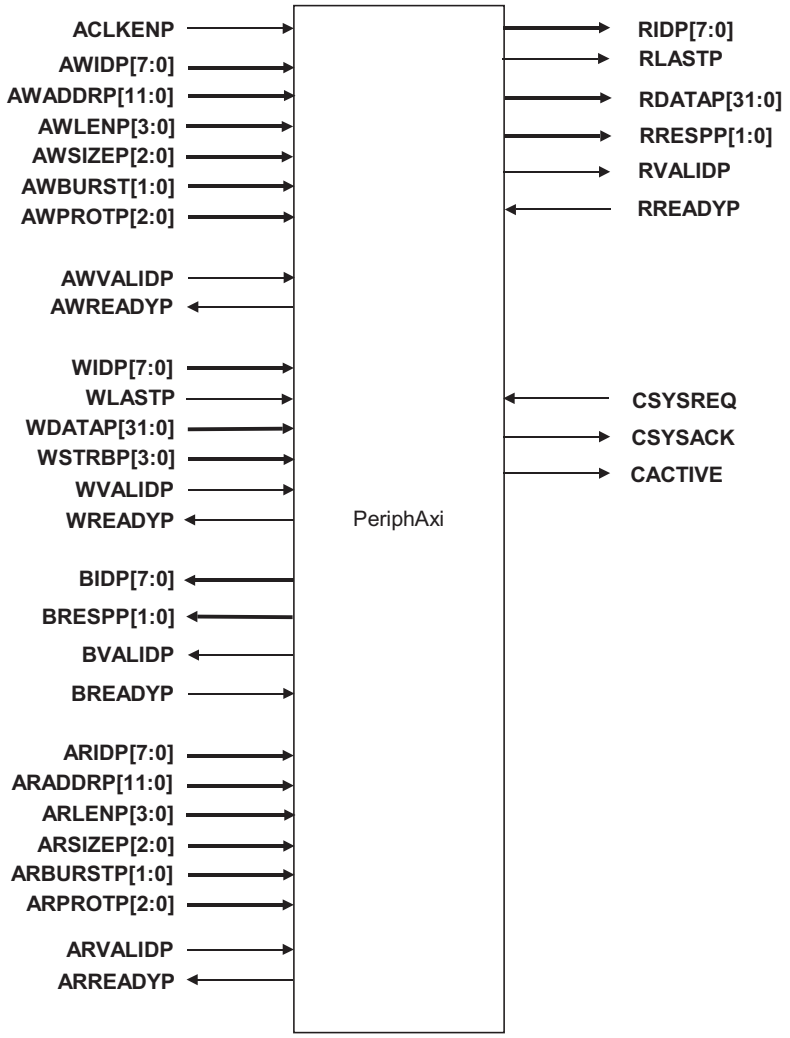


Figure 2-2 AXI peripheral port interface

2.1.4 TrustZone support

You can insert the cache controller into a system supporting TrustZone without modifications to the system. It ensures that NS transfers cannot read or modify secure data in the cache, and ensures that NS data cannot be evicted from the cache as a secure access.

See *TrustZone support in the cache controller* on page 2-25 for more information.

2.1.5 AXI low-power interface

The cache controller supports an AXI low-power interface, providing support for system clock gating. This is implemented by three signals, **CACTIVE**, **CSYSREQ** and **CSYSACK**.

CACTIVE indicates to the clock controller when the cache controller can be enabled. If HIGH the cache controller is processing a transaction, if LOW the cache controller is inactive, indicating that it can be disabled.

Note

CACTIVE has the inverse functionality of the **IDLE** signal on the L210 Cache Controller.

CSYSREQ is an input to the cache controller and generates requests for the cache controller to enter low-power state.

CSYSACK is used to acknowledge both the low-power state request and the exit from low-power state. This output is registered and is derived by the **CSYSREQ** line and the internal value of the **CACTIVE** signal.

Note

CSYSACK only indicates entry into low-power state when **CACTIVE** and **CSYSREQ** signals are both LOW.

See also *Low-power interface* on page 2-23 for more information.

2.1.6 Internal registers

The cache controller is controlled through a set of memory-mapped registers that occupy a relocatable 4KB area of memory. The registers can be written using the 32-bit AXI peripheral port.

See Chapter 3 *Programmer's Model* for more information.

2.1.7 Buffers

There are five main buffers:

- *Write buffer* on page 2-7
- *Write-allocate buffer* on page 2-7
- *Eviction buffer* on page 2-7
- *Line fill buffer* on page 2-7
- *Line read buffer* on page 2-7.

Write buffer

The WB has two slots that each contain a 256-bit data line and its associated address and security tag bit. The WB has merging capabilities. This means that successive writes to a same line address are merged in the same buffer slot. Any merging condition is based on address and on the security attribute **AWPROTS0[1]** or **AWPROTS1[1]**. The merging only takes place when data is in the WB and it has not been drained.

See Figure 2-1 on page 2-2 and *WB operation* on page 2-30 for more information.

Write-allocate buffer

The WAB permits allocation to the cache in the case where a write-allocate transaction is not performed directly by the WB. The data, address, security, and byte-valid information are sent to the WAB, when a WB slot is drained while its L2 memory attributes correspond to a write-allocate region, or write allocate behavior is forced by a corresponding control bit in the Auxiliary Configuration Register.

See Figure 2-1 on page 2-2 and *WAB operation* on page 2-31 for more information.

Eviction buffer

The EB is used for holding write-back data for L2 cache line evictions or cleaning of dirty cache lines. It holds 2 halves of a L2 cache line, 32 bytes total, and two address entries with their associated security attributes.

See Figure 2-1 on page 2-2 for more information.

Line fill buffer

The LFB is found on the master side of the cache and is used on a cache miss where an access to L3 is required. It is filled with data so that an entire 256-bit cache line can be allocated to the cache. If the original transaction is smaller than 256 bits the LFB is filled with neighboring locations in L3.

See Figure 2-1 on page 2-2 and *TrustZone support for the cache controller registers* on page 2-26 for more information.

Line read buffer

The 256 bit LRB is located on the slave side of the cache controller and is used during a read from the cache. A LRB is loaded with the complete cache line of the requested location to take advantage of spatial location for more read requests.

See Figure 2-1 on page 2-2 for more information.

Note

In an exclusive cache configuration where a cache entry is invalidated on a read hit, if the line also exists in the line read buffer, this is also invalidated.

2.1.8 Data RAM

The support of an 8-way L2 cache is based on increased hit rate and increased effectiveness of lockdown format C. With lockdown format C, a block of the L2 cache can be used as a *Tightly Coupled Memory* (TCM). The effective size of the L2 cache can be from 16KB to 2MB.

See *Data RAM configuration* on page 2-40 for more information.

2.1.9 Tag RAM

There is one tag RAM for each way of the L2 cache. A tag RAM is organized as a 21-bit wide memory. 18 bits are dedicated to the address Tag, one for security information and one for valid information and one for parity.

See *Tag RAM* on page 2-46 for more information.

2.1.10 Dirty RAM

The dirty RAM is used to identify if a line requires evicting when allocating new data to the cache. It is written in 1 or 2 bit writes for dirty writes or 16 bit reads for write-back evictions and cache maintenance operations. The dirty RAM is addressed using the same address bus as the tag RAM.

See *Dirty RAM* on page 2-44 for more information.

2.2 Functional operation

The function of the cache controller is to improve the performance of ARM-based systems when heavy traffic is generated from the ARM processor.

This section describes:

- *AXI master and slave interfaces*
- *AXI transaction ordering* on page 2-21
- *Exported AXI control* on page 2-21
- *AXI peripheral port interface operation* on page 2-22
- *TrustZone support in the cache controller* on page 2-25
- *Implementation details* on page 2-29
- *Data RAM configuration* on page 2-40
- *RAM interfaces* on page 2-42.

2.2.1 AXI master and slave interfaces

The masters and slaves are AXI-compatible. The cache controller can only support one outstanding read and one outstanding write from L1 to it each of its slave ports. It can only generate one outstanding read and one outstanding write from each of its master ports.

Table 2-1 shows the AXI master interface attributes.

Table 2-1 AXI master interface attributes

Attribute	Format
Write Issuing Capability	1
Read Issuing Capability	1
Combined Issuing Capability	2
Write ID Capability	1
Write Interleave Capability	1
Write ID Width	1
Read ID Capability	1
Read ID Width	8

Table 2-2 shows the AXI slave interface attributes

Table 2-2 AXI slave interface attributes

Attribute	Format
Write acceptance capability	1
Read acceptance capability	1
Combined acceptance capability	2
Write interleave depth	1
Read data reorder depth	1

This section describes:

- *ARM1176JZ(F)-S cache support*
- *Behavior for ARMv6 transactions* on page 2-12
- *Locked and exclusive accesses* on page 2-17
- *Master and slave port IDs* on page 2-19
- *Line fill buffer* on page 2-7
- *Line read buffer* on page 2-7.

ARM1176JZ(F)-S cache support

Table 2-3 describes the **AWCACHE[3:0]** and **ARCACHE[3:0]** signals as described in the AMBA AXI Protocol Specification, together with the ARMv6 equivalent.

Table 2-3 AWCACHE and ARCACHE definitions

AWCACHE / ARCACHE				AXI description	ARMv6 equivalent
WA	RA	C	B		
0	0	0	0	Noncacheable, nonbufferable	Strongly ordered
0	0	0	1	Bufferable only	Shared and non-shared device
0	0	1	0	Cacheable but do not allocate	Outer noncacheable
0	0	1	1	Cacheable and bufferable, do not allocate	-
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	-

Table 2-3 AWCACHE and ARCACHE definitions (continued)

AWCACHE / ARCACHE				AXI description	ARMv6 equivalent
WA	RA	C	B		
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, write allocate

Note

The shared input **AWSIDEBANDSx[0]** and **ARSIDEBANDSx[0]** is not described in this table, but the behavior is explained later in *Behavior for ARMv6 transactions* on page 2-12, under *Shared attribute*.

In this table, the bits have the following meaning:

B bit Bufferable. This means that the cache can delay the operation, and is only relevant for writes. The response sent to the processor comes from the cache controller, not from the final destination. In this case, any error response is forwarded to the processor by returning the L3 response if the transaction is non-secure.

C bit Cacheable. This means that the transaction at the final destination does not have to match the characteristics of the original transaction. For writes, this means that a number of different writes can be merged together. For read, this means that a location can be pre-fetched or can be fetched once for multiple transactions

RA bit Read Allocate. This means that the read transfer, if it misses, must be allocated in the cache.

WA bit Write Allocate. This means that the write transfer, if it misses, must be allocated in the cache.

Note

- The cache controller supports all AXI modes, even if the ARM1176JZ(F)-S processor does not use all of them
- If the cache controller receives cacheable fixed transactions, **AWBURSTSx** or **ARBURSTSx** = 2'b00, the results are unpredictable.

Behavior for ARMv6 transactions

Table 2-4 shows the general behavior of the cache controller depending on v6 transactions.

Table 2-4 Operational behavior descriptions

AXI description	ARMv6 transaction	Cache controller behavior
Noncacheable, nonbufferable	Strongly ordered	Read: Not cached in L2, results in L3 access Write: Not buffered, results in L3 access.
Bufferable only	Device	
Cacheable but do not allocate	Outer non cacheable	Read: Not cached in L2, results in L3 access Write: Put in WB, write to L3 when WB is drained.
Cacheable and bufferable but do not allocate		
Cacheable write-through, allocate on read	Outer write-through, no write allocate	Read hit: Read from L2 Read miss: Linefill to L2 Write hit: Put in WB, write to L2 and L3 when WB is drained Write miss: Put in WB, write to L3 when WB is drained.
Cacheable write-back, allocate on read	Outer write-back, no write allocate	Read hit: Read from L2 Read miss: Linefill to L2 Write hit: Put in WB, write to L2 when WB is drained, mark line as dirty Write miss: Put in WB, write to L3 when WB is drained.
Cacheable write-through, allocate on write		Read hit: Read from L2 Read miss: not cached in L2, causes L3 access. Write hit: Put in WB, write to L2 when WB is drained Write miss: Put in WB, put in WAB if not full when WB is drained, data request for word/line to L3, allocation to L2, write to L3.
Cacheable write-back, allocate on write		Read hit: Read from L2 Read miss: not cached in L2, causes L3 access. Write hit: Put in WB, write to the L2 when WB is drained, mark line as dirty Write miss: Put in WB, put in WAB if not full when WB is drained, data request for word/line to L3, allocation to L2.

Table 2-4 Operational behavior descriptions (continued)

AXI description	ARMv6 transaction	Cache controller behavior
Cacheable write-through, allocate on read and write	Outer write-through, no write allocate with L220 write_allocate_override bit set	Read hit: Read from L2 Read miss: Linefill to L2 Write hit: Put in WB, write to L2 and L3 when WB is drained Write miss: Put in WB, put in WAB if not full when WB is drained, data request for word/line to L3, allocation to L2, write to L3.
Cacheable write-back, allocate on read and write	Outer write-back, write allocate	Read hit: Read from L2 Read miss: Linefill to L2 Write hit: Put in WB, write to the L2 when WB is drained, mark line as dirty Write miss: Put in WB, put in WAB if not full when WB is drained, data request for word/line to L3, allocation to L2.

Noncacheable transactions are not processed by the tag lookup logic.

Note

On writeback allocations or writeback hits to the cache, the line might not be marked as dirty if the **WRITEBACKSx[1:0]** signals indicate that the transaction is an eviction and it has clean status.

Other behaviors are described in:

- *Shared attribute*
- *Force write allocate* on page 2-14
- *Exclusive cache configuration* on page 2-15.

Shared attribute

This section describes transactions through the **AWSIDEBANDS1**, **ARSIDEBANDS1**, **AWSIDEBANDS0**, and **ARSIDEBANDS0** signals.

The processor can specify that transactions are shared through the **ARSIDEBANDSx** or **AWSIDEBANDSx** signals, by using bit [0] only. Where bit[0] is configured to the binary value of 1’b0 the transaction is non-shared, and if configured to 1’b1 the transaction is treated as shared.

To preserve coherency, no allocation can happen on shared transfers, and no lookup is requested on the internal cache controller. However, the write transfers are merged if they are bufferable, that is to say if the C bit of **AxCACHESx** is set.

Setting the shared attribute override bit [22] in the Auxiliary Control Register causes a shared access to be internally treated as non-shared in the Cache Controller.

Note

Dynamically changing the shared override bit without flushing the cache could cause a hazard where incorrect data could be evicted causing more recent data in L3 to be overwritten.

Force write allocate

The default setting for the Force write allocate bits [24:23] is 2'b00 in the Auxiliary Control Register at `BASE + 104`, and this configures the cache controller to use the received **AWCACHE** or **ARCACHE** attributes. Additional reference data on the general behavior can be found in Table 2-3 on page 2-10 and Table 2-4 on page 2-12.

Note

DECERR is a decode error.

If you set the Force write allocate bits to 2'b01 in the Auxiliary Control Register, this causes the WA bit to be always set to 1'b0. No write allocation is performed, so the data is always checked with L3 for security.

If you set the Force write allocate bits to 2'b10 in the Auxiliary Control Register, this causes cacheable write misses to be allocated in the cache. With reference to the ARM1176JZF-S processor, the changes are the following:

- Cacheable write-through, allocate on the read transaction attribute:
  - For a write miss instruction, when the WB is drained, the data is sent to L3 and written in the WAB. If the line is not full, a linefill is performed and merged with the data.
- The cacheable write-back, allocate on the read transaction attribute is treated as a cacheable write-back allocate on read and write.

Note

- The **AWSIDEBANDSx[0]** signal takes priority over the force write allocate settings, that is, if **AWSIDEBANDSx[0]** is set it causes the transaction to be non-cacheable.

- The Force Write Allocate configurations do not change the **AWCACHE/ARCACHE** attributes present on the master ports to L3.

Exclusive cache configuration

Exclusive cache configuration is a new feature for r1p0. Setting bit [12] to 1'b1 in the Auxiliary Control Register causes the Cache Controller to change its caching policy by reading the following signals:

- **AxPROTSx[2]**
- **AxSIDEBANDSx[4:1]**, inner cache attributes
- **WRITEBACKSx[1:0]**
- **AxCACHESx**, provided by the L1 on a transaction basis.

The exclusive cache configuration changes the caching policy for data transactions only. The cache policy for instruction transactions is unchanged. When instruction reads are detected there is no more requirement to read the values of **ARSIDEBANDSx** or **WRITEBACKSx[1:0]**. For instructions the caching policy is derived by reading **ARCACHESx** only.

The exclusive cache configuration does not permit data to reside in L1 and L2 at the same time. The exclusive cache configuration provides support for only caching data on an eviction from L1 when the inner cache attributes are write-back, cacheable and allocated in L1. The key behavior is summarized as follows:

- When data is requested by L1 for allocation and results in a L2 hit, the data is passed to L1 and the L2 cache line is cleaned and invalidated.
- When data is requested by L1 for allocation and results in a L2 miss, the **ARCACHESx** attributes are overridden, L2 allocation is not performed and the data is passed to L1 without allocation at L2.
- When data is written by L1 with the **WRITEBACKSx[0]** signal set to one, which indicates an eviction, then the write access is treated as follows:
  - If **AWCACHESx** is write-through, data is not allocated and L3 is written to.
  - If **AWCACHESx** is write-back and the access hits in the L2 cache, the behavior depends on the value of **WRITEBACKSx[1]**. If WRITEBACKSx[1] is set to zero, the data is written to the cache and marked as dirty. If WRITEBACKSx[1] is set to one, no action is required.
  - If **AWCACHESx** is write-back and the access misses in the L2 cache, the behavior depends on the value of **WRITEBACKSx[1]**. If WRITEBACKSx[1] is set to zero, the data is allocated into the cache and marked as dirty. If WRITEBACKSx[1] is set to one, the data is allocated into the cache and marked as clean.

Note

- The Cache Controller only reads the value of **WRITEBACKSx[1]** when **WRITEBACKSx[0]** is set, otherwise this value is ignored.
- The Cache Controller uses the **WRITEBACKSx[1:0]** signals regardless of the value of the exclusive cache configuration bit [12] in the Auxiliary Control Register. The L1 master cannot issue a write burst where the write data consists of a mix of dirty and clean data.
- You must tie LOW the **WRITEBACKSx** signals in the case where the Cache Controller is connected to a processor that does not use them.

Table 2-5 indicates the changes in caching policy when an exclusive cache configuration is enabled.

Table 2-5 Caching policy changes when exclusive cache is enabled

Inner cache attributes (AxSIDEBANDSx)	Outer cache attributes (AxCACHESx)	L2 Write Miss + WRITEBACKSx[0] set	Read hit	Read miss
WBRA	WTRA	allocate + L3 (Wr)	clean + invalidate	no allocation
WBRA	WBRA	allocate	clean + invalidate	no allocation
WBRA	WTWA	allocate + L3 (Wr)	clean + invalidate	no allocation
WBRA	WBWA	allocate	clean + invalidate	no allocation
WBRA	WTRAWA	allocate + L3 (Wr)	clean + invalidate	no allocation
WBRA	WBRAWA	allocate	clean + invalidate	no allocation
WBRAWA	WTRA	allocate + L3 (Wr)	clean + invalidate	no allocation
WBRAWA	WBRA	allocate	clean + invalidate	no allocation
WBRAWA	WTWA	allocate + L3 (Wr)	clean + invalidate	no allocation
WBRAWA	WBWA	allocate	clean + invalidate	no allocation
WBRAWA	WTRAWA	allocate + L3 (Wr)	clean + invalidate	no allocation
WBRAWA	WBRAWA	allocate	clean + invalidate	no allocation

WTRA Writethrough with read allocate set.
WTWA Writethrough with write allocate set.
WTRAWA Writethrough with read and write allocate set.

WBRA Writeback with read allocate set.
WBWA Writeback with write allocate set.
WBRAWA Writeback with read and write allocate set.

Inferring an additional cycle of latency

Because of the invalidation of the cache line on a read hit when the exclusive cache configuration is enabled, it is possible that the cache controller can infer an additional cycle of latency.

Table 2-6 shows how this can be demonstrated.

Table 2-6 Differences of cache controller pipeline behavior for exclusive configuration

	Cycle 1	Cycle 2	Cycle 3	Cycle 4
Op1 exclusive configuration off	TR	DR		
Op2 exclusive configuration off		TR	DR	
Op1 exclusive configuration on	TR	DR + TW		
Op2 exclusive configuration on		STALL (Op1 TW)	TR	DR

Where.
TR indicates a tag read or a lookup
DR indicates a data memory read
TW indicates a tag write, to write to the invalid field.

Locked and exclusive accesses

These are as follows:

Locked transfers

For a locked access, the **AWCACHE** or **ARCACHE**, information can be used depending on whether the transaction is read or write. In the case of a noncacheable transfer, the access is forwarded to L3 through the master ports and is marked as locked. No cache lookups are performed in this case.

Note

Noncacheable transfers are noncacheable, nonbufferable, bufferable only, normal non-shared, normal shared or cacheable that do not allocate.

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

Note

- Cacheable transfers are normal non-shared or normal shared with shared attribute override, except cacheable accesses that do not allocate for both read or write.
- Write accesses are write-through or write-back miss.

When a slave is performing a locked transfer, cacheable or not, the other slave is stopped from accepting more transfers. A locked transaction is stalled if there are outstanding transactions in either master port or either slave port.

If a locked transfer is issued to **S0** and **S1**, the peripheral port accepts a background operation but stalls it until the locked transfer is complete.

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

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

Note

A locked sequence must consist of solely noncacheable or cacheable transactions. A locked sequence cannot contain a mix of cacheable and noncacheable transactions.

Exclusive accesses

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

The Cache Controller supports cacheable and noncacheable exclusive accesses.

On the ARM1176JZ(F)-S processor, all exclusive transfers are marked as shared. In the Cache Controller the shared bit is propagated to L3, but the transfer is never cached whatever the **AWCACHE** or **ARCACHE** values are. However, the shared attribute override bit in the Auxiliary Control Register changes these transfers to cacheable and bufferable.

If the shared override functionality is used, it is the responsibility of the system integrator to implement an external exclusive monitor, so that the EX OKAY response can be returned.

Note

- An L3 exclusive monitor must be able to handle an exclusive read and write from the same master at the same time.
- If you require an exclusive access on cacheable data then you require one or more exclusive monitors on the slave side of the Cache Controller.
- If you require an exclusive access on noncacheable data then you require one or more exclusive monitors on the master side of the Cache Controller.

Note

Non-cacheable locked and exclusive accesses are also described in *AXI transaction ordering* on page 2-21.

Master and slave port IDs

The master and slaves are AXI-compatible. 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**
- Peripheral Port: 8 bits [7:0], signals **AWIDP**, **ARIDP**, **WIDP**, **BIDP**, and **RIDP**
- 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**.

The master port ID bus, namely **AWIDMx**, **ARIDMx** and **WIDMx**, is made up by the level 1 transaction ID of 6 bits, namely **ARIDSx**, **AWIDSx**, **WIDSx**, from the slave port.

Note

The **x** in a signal name, for example **AWIDMx** indicates that a 0 or 1 can be substituted indicating two actual signals, that corresponds to the respective port.

Where the Cache Controller is configured to merge **S0** and **S1** slave ports, Bit [0] of the **M1** ID is used to identify the slave port that the transfer originated from. This bit is set to a binary value of 1’b0 for noncacheable locks or exclusive transfers requested by slave 0 and set to a value 1’b1 for noncacheable lock or exclusive transfers requested by slave 1.

Also, in the case of each master port, Bit[7] is reserved for future use of the Cache Controller. This bit is statically set to 1’b0 for all transactions.

Table 2-7 shows the format of the identifications that are exported on the Master Ports.

Table 2-7 Master Port ID values

Master ID Buses	Comment	ID Value (Verilog)
ARIDM1, **AWIDM1**, **WIDM1**	Noncacheable Exclusive or Lock Read/Write transaction that originates from slave 0 port	{1’b0,L1_ID, 1’b0}
ARIDM1, **AWIDM1**, **WIDM1**	Noncacheable Exclusive or Lock Read/Write transaction that originates from slave 1 port	{1’b0,L1_ID,1’b1}
ARIDM1, AWIDM1, WIDM1	All other transactions	{8’b0000_0000}
ARIDM0, AWIDM0, WIDM0	Noncacheable Exclusive or Lock Read/Write transaction that originates from slave 0 port Port **M0** cannot receive noncacheable transactions from slave port 1, so for noncacheable exclusives or locks **M0** always drives 1'b0 on bit[0]	{1’b0,L1_ID,1’b0}
ARIDM0, AWIDM0, WIDM0	All other transactions	{8’b0000_0000}

Note

L1_ID = Level 1 Master, transaction ID, maximum width of 6-bits.

These ID bits are returned with the data on the RID lines in the case of a read and accompany the write response signal as BID in the case of a write.

You must consider the following, if the ID bits are static or ID bits are not supported by the L1 master:

- When connecting a master to the Cache Controller, the *System on Chip* (SoC) integrator must tie off the buses **ARID**, **AWID**, and **WID**. Then **RID** and **BID** are left as floating outputs from the L2 cache controller.
- When connecting to a SoC on the Cache Controller Master Side, use the L3 SoC interconnect to tie off **RID[7:1]**, **BID[7:1]** and let **ARID[7:1]**, **AWID[7:1]** and **WID[7:1]** remain as floating outputs.

2.2.2 AXI transaction ordering

This section describes AXI transaction ordering through the master port.

Master port

The cache controller only issues a transaction ID of 8'b00000000 from the master port, unless the transaction is a noncacheable exclusive or a noncacheable lock. In either of these cases the cache controller propagates the transaction ID that the L1 master issues.

The cache controller stalls exclusive and lock transactions until the there are no outstanding transactions waiting on the master port. Consequently, when a lock or exclusive transaction is in process it is not possible to issue any more outstanding transactions on the respective master until a response and data has been returned from the lock or exclusive. Issuing multiple outstanding transactions with different IDs is not supported with the cache controller design, because this could warrant the return of interleaved read data downstream in L3. The cache controller cannot support the return of interleaved read data, because it only has a LFB depth of one for each master port.

2.2.3 Exported AXI control

Table 2-8 provides information on the AXI control signals that are exported on the master ports of the cache controller.

Table 2-8 Exported master ports AXI control signals

Access type	Master control signals buses	Value (Verilog)
Noncacheable read transactions, nonbufferable write transactions, or cache disabled	**AWCACHEMx, ARCACHEMx** **AWSIDEBANDMx, ARSIDEBANDMx** **AWPROTMx, ARPROTMx** **AWLOCKMx, ARLOCKMx**	all as original transaction
Linefills associated with a RA	**ARCACHEMx**	as original transaction
	ARSIDEBANDMx	{5'b00000}
	ARPROTMx	as original transaction
	ARLOCKMx	{2'b00}

Table 2-8 Exported master ports AXI control signals (continued)

Access type	Master control signals buses	Value (Verilog)
WA Linefill	**ARCACHEM1**	{4'b1010}
	ARSIDEBANDM1	{5'b00000}
	ARPROTM1	{1'b0,SECURITY,1'b1}
	ARLOCKM1	{2'b00}
All bufferable write transactions	**AWCACHEM1**	{4'b0010}
	AWSIDEBANDM1	{5'b00000}
	AWPROTM1	{1'b0,SECURITY,1'b1}
	AWLOCKM1	{2'b00}
Evictions	**AWCACHEM1**	{4'b0011}
	AWSIDEBANDM1	{5'b00000}
	AWPROTM1	{1'b0,SECURITY,1'b1}
	AWLOCKM1	{2'b00}

Note

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

2.2.4 AXI peripheral port interface operation

The peripheral port interface consists of the following operational components:

- *Address channels* on page 2-23
- *Read and write channels* on page 2-23 on page 2-23
- *Write response channel* on page 2-23
- *Low-power interface* on page 2-23
- *Peripheral AXI error response* on page 2-24
- *Address encoding* on page 2-25
- *Exclusive access support* on page 2-25.

Address channels

The peripheral port asserts **AWREADYP** until **AWVALIDP** is active and similarly asserts **ARREADY** until **ARVALIDP** is active, at that point the burst address and control information is accepted. The Address channels are waited, by driving **AWREADYP** or **ARREADYP** LOW. This occurs from when this valid address is accepted until the last transfer in the burst is valid.

Read and write channels

The read and write channels are waited, by driving **WREADYP** or **RVALIDP** respectively LOW, until a burst address is accepted.

The last transfer is indicated for write transfers by **WLASTP** being asserted. An internal counter tracks the number of read transfers in the burst. For read transfers, the peripheral port indicates the last transfer in the burst by asserting **RLASTP**.

Write response channel

BVALIDP is deasserted until the last transfer in the write burst has completed. **BVALIDP** is then asserted and the burst write response is issued on **BRESPP** until **BREADYP** is asserted, indicating that the interconnect has accepted the response.

Low-power interface

The output signal **CACTIVE** indicates when the cache controller is not idle and is doing any internal processing.

If the L1 processor has entered a wait for interrupt state, and the cache controller is now idle, that is, **CACTIVE** is not asserted, the clock to both the L1 processor and cache controller can then be gated. The cache controller clocks must be resumed at the same time or before the L1 processor, so that any AXI requests from the L1 processor can be handled by the cache controller.

Note

- The Cache Controller **CACTIVE** signal can be LOW even if the WB is not empty.
- The Cache Controller **CACTIVE** signal has the inverse functionality of the L210 **IDLE** signal.

Peripheral AXI error response

The slave response is encoded according to AMBA AXI Protocol. The error response signals are described in:

- *Write response*
- *Read response*.

Write response

If the AXI peripheral interface receives a NS write access instruction to a secure register, the data response results in a write response equal to a DECERR.

If the AXI peripheral interface receives a secure write access to a secure register or a NS register, or a NS write access to NS data, the following response is given:

- A write data instruction response of **BRESPP** is OKAY, but only if you attempt a supported write access to any address within the 4KB slave memory map, including the addresses occupied by the read/write registers or their aliases.

A SLVERR response is returned to the master if the AXI peripheral interface receives any of the following unsupported accesses:

- Any accesses with **AWSIZE** information other than 32-bit receives a SLVERR response.
- Any accesses with **AWLEN** information other than zero receives a SLVERR response.
- Any access that is unaligned, for example, where **AWADDRP[1:0]** is not equal to 2'b00, returns a SLVERR response where a read access returns all zeros and a write access does not modify the address location.
- Any write access that attempts to make use of the **WSTRB** lines, for example where any bits of **WSTRB[3:0]** are 0, returns a SLVERR response and does not modify the address location.

Read response

If the AXI peripheral interface receives a NS read access to a secure register, the read response results in DECERR. The following registers are enabled to read with NS accesses, any NS read instructions return valid data and an OKAY response.

- L2 Control Register (`0x100`)
- L2 Auxiliary Control Register (`0x104`)
- L2 Cache Lockdown (`0x900` and `0x904`), the default behavior, can be overridden for NS Read/Write.

If the AXI peripheral interface receives a secure read access to a secure register or a NS register, or a NS read access to NS data then a read data response **RRESPP** is OKAY. If a supported read access is made to any address within the 4KB slave memory map, and access is made in a undefined memory space then zeros are returned as data. Also:

- any accesses with **ARSIZE** information other than the 32-bit receives a SLVERR response, where a read access returns all zeros
- Any accesses with **ARLEN** information other than zero receives a SLVERR response.
- on unaligned read bursts, the slave returns **RDATA** = 0 and a SLVERR response for the every beat of the burst.

Address encoding

Table 2-9 shows the address encoding bits.

Table 2-9 Address encoding bits

Bit	Function
[31:12]	Not used
[11:8]	Register selection
[7:4]	Register block selection
[3:0]	Word selection

Exclusive access support

Exclusive access is not supported. An exclusive access to the AXI peripheral port interface returns an OKAY, exclusive access fail, response.

2.2.5 TrustZone support in the cache controller

Some aspects of TrustZone support for the cache controller are as follows:

- A NS tag bit is attached to any data in the cache or in buffers. It is not possible to access secure data with a NS access.

- Trying to access data in the cache that is marked secure with a NS access is treated as a miss. For a read transfer, a linefill command is sent to L3 memory, and any security error is propagated to the processor. The security error generated is dependent on the security mechanism employed by the L3 memory. This can enable secure transfers to access NS memory.
- To enable or disable the L2 Control Register, you can write to it with an access tagged as secure.
- You can only write to the Auxiliary Control Register with an access tagged as secure.
- NS maintenance operations do not clean and/or invalidate secure data.
- Bit [26] in the Auxiliary Control Register is for NS Lockdown enable and can only be modified with secure accesses. The bit is used to tell whether a lockdown register can be modified by NS accesses.
- You can only perform test operations by secure accesses, it is not possible to read or write secure or NS data with NS accesses to the test registers.

TrustZone support is described in:

- *TrustZone support*
- *TrustZone support for the cache controller registers*.

TrustZone support

See *ARM Architectural Reference Manual, Security Extensions supplement* for more information on how Trustzone is supported by the system.

TrustZone support for the cache controller registers

In a system supporting Trustzone security extensions, you must map the cache controller registers as secure within the secure page tables. The same registers are marked as NS within the NS page tables. This enables the cache controller registers that are secure-only to be accessed by secure processes.

A NS write access, **AWPROT[1]** = 1, to any of the secure registers returns a BRESP response of DECERR, where the data is not written. The NS world maps the cache controller registers as NS and are only be able to access the unrestricted registers.

Linefill mechanism

In a cacheable secure read or write allocate miss, the linefill is requested using secure access. The data is tagged in the cache as secure data.

In a cacheable NS read or write allocate miss, the linefill is requested using NS access. If no security error is received, the data is tagged in cache as NS data, otherwise no allocation is performed, and the data is lost for the non-permitted WA instruction. The L1 master is returned a BRESP response of DECERR for the write allocate instruction and a RRESP response of DECERR for the read allocate instruction.

System security aborts

External aborts to the processor are essentially fatal errors and are normally extremely rare. In the case of a secure system, external aborts can be caused by a mismatch between the secure protection signal and the level of protection of the targeted memory.

If an access to L3 memory causes an AXI DECERR response the requesting processor aborts.

Secure access to a NS area can also return an OKAY response.

2.2.6 Power modes

Power modes are controlled by clock/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 any of the following:

- *Run mode*
- *Standby mode*
- *Dormant mode* on page 2-28
- *Shutdown mode* on page 2-29.

Run mode

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

Standby mode

Standby mode disables most of the clocks of the device, while keeping the design powered up. This reduces the power drawn to the static leakage current, plus a small amount of clock power overhead is required to enabled the device to wake up from the standby state.

Ensure that the cache controller goes into standby mode, when the L1 masters have been put into standby wait for interrupt mode. You must ensure that the cache controller returns to run mode prior to returning active clock edges to the L1 masters.

On ARM11 processors, entry into standby mode is performed by executing the Wait For Interrupt CP15 operation. The transition from standby mode to run mode is caused by any of the following:

- the arrival of an interrupt, whether masked or unmasked
- an imprecise external abort, whether masked or unmasked
- a debug request, whether debug is enabled or disabled
- reset.

Before entering L2 standby mode, you must perform an L2 Cache Sync operation to ensure the memory system is not affected by the entry of the standby state. Check that the system power/clock controller interrogates the cache controller AXI low-power interface to ensure that the cache controller is idle.

Dormant mode

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

Because the write buffer and the internal register banks are implemented in standard cells, it is necessary for you to perform a Cache Sync operation, that causes write buffer drain. Also you must save the values of the programmed registers before power can be removed from the cache controller. You must therefore perform the following software operations before entering dormant mode:

1. Disable the cache controller with a write to the Control Register, `0x100`, this sequence performs the Cache Sync operation.
2. Poll the Control Register, `0x100`, until the cache is disabled.
3. Read and save the state of the Auxiliary Control Register, `0x104`. Also if required, do the same to the event/interrupt and lockdown registers.
4. Start **CSYSREQ/CACTIVE/CSYSACK** handshaking.
5. Remove power.
6. Restore power.
7. Reset the cache controller.
8. Complete **CSYSREQ/CACTIVE/CSYSACK** handshaking.
9. Write the saved state of the Auxiliary Control Register, `0x104`, back to the register. Also if required do the same to the event/interrupt and lockdown registers.
10. Enable the cache with a write to the Control Register, `0x100`.

11. Poll the Control Register, `0x100`, until the cache is enabled.

As with standby mode, the cache controller can go into dormant mode, only when the L1 masters are inactive and issuing no more transactions. Ensure the cache controller is placed back into run mode prior to returning active clock edges to the L1 masters.

Dormant mode considerations

Dormant mode is only partially supported on the cache controller, because care is required in implementing this on a standard synthesizable flow. The RAM blocks that require powering up must be implemented on a separate power domain. You must also clamp all of the inputs to the RAMs to a known logic level, with the chip enable being held inactive. You must not implement clamping in gates as part of the default synthesis flow because it contributes to a critical path. The **RAMCLAMP** input is provided to drive this clamping.

You must add clamps to the placeholder, either as explicit gates in the RAM power domain, or as pull-down transistors that clamp the values when the cache controller is powered down.

If dormant mode, is implemented, the following RAM blocks must remain powered up:

- all L2 cache data RAMs
- all L2 cache data parity RAMs, if implemented
- all L2 cache tag RAMs
- all L2 cache dirty RAMs.

Transition from dormant state to run state is triggered by the external power controller asserting reset to the cache controller until the power is restored. When the power has been restored to the cache controller it leaves reset. The processor then interrogates the external power controller, to determine whether the saved state can be restored. See the ARM11 TRMs for more information.

Shutdown mode

Shutdown mode has the entire device powered down. 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.

2.2.7 Implementation details

This section is describes:

- *Disabled operation* on page 2-30
- *WB operation* on page 2-30
- *WAB operation* on page 2-31

Disabled operation

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

WB operation

Two buffered write accesses to the same address and the same security bit causes the first one to be overridden if the WB has not been drained in between.

The WB has a merging capability. This means that lines are not dealt with as soon as they contain data. The WB draining policy is:

- A WB slot is drained when it is full.
- WB is drained at each noncacheable read occurrence.
- WB is drained at locked or exclusive nonbufferable write occurrences.
- If the two slots of the WB contain data, the least recently accessed is drained.
- If a hazard is detected with one WB slot, it is drained to resolve the hazard

The WB has two slots that each contain a 256-bit data line, its associated address and security tag bit. Each slot contains a byte-valid field that enables the control logic to determine the data line fill level. If a drained slot is drained while its data line is not full, it is the responsibility of the WB to request correct transactions to the master port.

For WA transactions, the WB transfers information from one of its slots to a WAB.

WAB operation

Based on byte-valid information, the WAB requests, through the **M1** master port to L3 main memory, the data required to fill its data line prior to allocation:

- If one to eight sequential bytes are missing, one single 64-bit transaction is performed on L3 main memory.
- In all other cases, a full linefill request is performed.

Data from the WB and the master port are then merged in the WAB this can then request an allocation to the cache.

If the master port receives an error response during read transfer for WAB, the allocation from WAB to the cache is not performed.

Cache configurability

Table 2-10 shows how the cache controller can be configured.

Table 2-10 Cache controller cache configurability

Feature	Enabled by	Range of options	Default value	Default option
Cache way size	Register	16KB, 32KB, 64KB, 128KB, 256KB	3'b001	16KB
Number of cache ways	Register	0, 1, 2, 3, 4, 5, 6, 7, 8	4'b0000	Associativity, cache absent
Latency	Register	1, 2, 3, 4, 5, 6, 7, 8	3'b111	8 cycles of latency
Event monitor bus	Register	Disabled (0) / Active (1)	0	Disabled
Number of master ports	Pin	1 / 2		Defined by **MASTNUM** pin
Number of slave ports	Pin	1 / 2		Defined by **SLAVENUM** pin

Note

If a single slave port (**AXI S1**) is configured it is expected that only a single master port (**AXI M1**) is configured. The Master port (**AXI M0**) is not active or supported when the cache controller is configured for **SLAVENUM** = 0.

Cache Controller ports configuration

You can completely remove the **M0** master port by the use of Verilog directives, see the *L220 Cache Controller Implementation Guide* for more information. To simplify AMBA port design, you can configure the Cache Controller. These configurations are:

Number of master ports

The cache controller can work with two **AXI M0** and **AXI M1** master ports or one **AXI M1** only master port. The choice is made by forcing a correct value on the **MASTNUM** input pin. The clock of an unused master is inactive in functional mode, and is made active in test mode when the **SCANENABLE** input pin is HIGH.

Note

If only one slave is configured it is expected that only one master is supported, a configuration of one slave and two masters is not a supported and tested Cache Controller configuration.

Number of slave ports

The Cache Controller can work with two **AXI S0** and **AXI S1** slave ports or one **AXI S1** only slave port. The choice is made by forcing a correct value on **SLAVENUM** input pin. The clock of an unused slave is inactive in functional mode and is made active in test mode when the **SCANENABLE** input pin is HIGH.

Note

If the system designer requires the peripheral port of the Cache Controller to be big-endian, then the WDATAP and RDATAP buses must be crosswired on a four-byte basis when connecting the cache controller. In this case, if the contents of the L2 cache are accessed using the test registers, the RAMs appear to be big-endian. The other Cache Controller slave and master ports do not require such cross-wiring to support big-endian accesses.

External error support for L3 memory

The Cache Controller gives limited support for external aborts, because of restrictions of the AXI protocol. However, there are methods to acknowledge all external aborts created by a slave device.

The key points are:

- The Cache Controller is a slave to the L1 master processor, but a master to the L3 main memory system. In effect this means that the Cache Controller is a method to communicate between the master processor and the slave L3 main memory.
- 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.

In the case where L3 detects a security mismatch it responds with a DECERR response.

External error is described as follows:

Write transactions

If the transfer is a noncacheable write, an error response is given by L3.

If the transfer is a cacheable 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 beat of data returned. In the case of an error returned for an RA line fill, the line is not loaded into the data RAM and the tag RAM is not updated.

Evictions or WB drain

If the request came from an eviction or WB drain to L3, then the write response error response cannot be passed back to L1 because the Cache Controller no longer has the transaction outstanding in the slave port.

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

It is recommended that you connect these interrupt sources through an interrupt controller to the L1 processor.

Error response summary

Table 2-11 shows error responses for all combinations of L3 access.

Table 2-11 Error responses for all combinations of L3 access

Access type	Error response mechanism
Nonlocked and nonexclusive noncacheable bufferable write transactions	Error response is imprecise and is passed back to the L1 processor using the interrupt lines.
All other noncacheable transactions	Error response is precise and is passed back to the L1 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 L1 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.
WA linefill	All error responses are imprecise and are passed back to the L1 processor using the interrupt lines. Data is not allocated.
Write-throughs	All error responses are imprecise and are passed back to the L1 processor using the interrupt lines. Data is not allocated.
Evictions	All error responses are imprecise and are passed back to the L1 processor using the interrupt lines.

Note

- If a write transaction is a NS write-through allocate, write or both, partial line, tag miss then this would generate a write-through and write allocate linefill in the master. In this case the Cache Controller passes the write response from the write-through response and not the linefill read response.
- Evictions, WA linefill, Write-throughs, RA linefills and some parity errors are the operations that utilize the **SLVERRINTR** or **DECERRINTR** interrupt lines.

Cache event monitoring

ARM supplies pins for event monitoring of the Cache Controller cache signals. 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 Debug Control Register bit[2]. You must perform a cache sync operation prior to debug or any analysis that relies on this signal. Synchronizers for the signal are added to prevent any issues if from asynchronous domain control.

The event monitoring bus is enabled by writing to bit [20] of the L220 Auxiliary Control Register. Table 2-12 shows the event pins.

Table 2-12 Event pins

L2CC pin	Description
CO	Eviction (CastOUT) of a line or a half line from the L2 cache.
DRHIT	Data read hit in the L2 cache.
DRREQ	Data read lookup to the L2 cache. Subsequently results in a hit or miss.
DWHIT	Data write hit in the L2 cache.
DWREQ	Data write lookup to the L2 cache. Subsequently results in a hit or miss.
DWTREQ	Data write lookup to the L2 cache with Write-Through attribute. Subsequently results in a hit or miss.
IRHIT	Instruction read hit in the L2 cache.
IRREQ	Instruction read lookup to the L2 cache. Subsequently results in a hit or miss.
SPNIDEN	Secure privileged non-invasive debug enable.
WA	Allocation into the L2 cache caused by a write (with Write-Allocate attribute) miss.

Cache interrupts outputs

Table 2-13 on page 2-36 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 2-13 Interrupts

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

Parity and RAM error support

Figure 2-3 on page 2-37 shows cache controller parity and RAM error support. The cache controller generates the parity write data for the data and tag RAMs. For:

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

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

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

In addition to parity error detection, there are error inputs on RAM interface, one from Data, **DATAERR** and eight from tag RAM, **TAGERR[7:0]**. You can use them to identify read and write errors from the RAMs. Those errors are treated in the same way as parity errors.

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

If a parity error occurs on tag or data RAM during AXI write transactions, a SLVERR error response is reported back to the L1 through an interrupt on the SLVERRINTR line.

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

Figure 2-3 shows the cache controller parity and RAM error support.

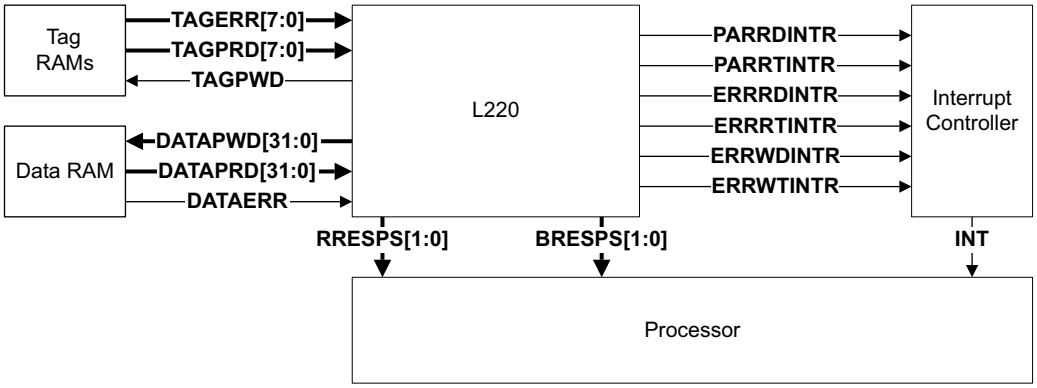


Figure 2-3 L220 parity and RAM error support

Error signalling and handling

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

- *Cacheable read requests on AXI slave ports* on page 2-38
- *Write access from the write buffer* on page 2-38
- *AXI master or write-allocate buffer allocation requests* on page 2-38
- *Clean maintenance* on page 2-39
- *Invalidate maintenance operation* on page 2-39
- *Clean and Invalidate maintenance operation* on page 2-40.

Cacheable read requests on AXI slave ports

Table 2-14 shows the error signalling cases and effects when there are cacheable read requests on the AXI salve ports.

Table 2-14 Cacheable read requests on AXI slave ports

Case	Effect
Tag parity or RAM error on cache lookup	An error is flagged through the **PARRTINTR/ERRRTINTR** interrupt signals, the values returned on **RDATASx** are zero.
Data RAM error on data write	An error is flagged through the **PARRTINTR/ERRRTINTR** interrupt signals, the values returned on **RDATASx** are zero.

Write access from the write buffer

Table 2-15 shows the error signalling cases and effects when there are write-through/write-back write accesses from the write buffer.

Table 2-15 Write-through/write-back write access from WB

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

AXI master or write-allocate buffer allocation requests

Table 2-16 shows the error signalling cases and effects when there are AXI master or write-allocate buffer allocation requests.

Table 2-16 AXI M0 and AXI M1 masters or write-allocate buffer allocation requests

Case	Effect
Tag parity or RAM error on cache lookup, for next victim selection	An error is flagged through the **PARRTINTR/ERRTINTR** interrupt signals, the allocation process is not stopped.
Data parity or RAM error on data read for eviction, all ways contain data to targeted index and next victim line is dirty	An error is flagged through the **PARRDINTR/ERRRDINTR** interrupt signals, the allocation process is not stopped but the eviction is not performed.

Table 2-16 AXI M0 and AXI M1 masters or write-allocate buffer allocation requests (continued)

Case	Effect
Tag RAM error on tag write	An error is flagged through the **ERRWTINTR** interrupt signal, subsequent cache lookups to the same index are unpredictable.
Data RAM error on data write	An error is flagged through the **ERRWDINTR** interrupt signal, subsequent reads to that line are unpredictable.

Clean maintenance

Table 2-17 shows the clean maintenance operation error signalling cases and effects.

Table 2-17 Clean maintenance operation cases

Case	Effect
Tag parity or RAM error on cache lookup	An error is flagged through the **PARRTINTR/ERRRTINTR** interrupt signal, the clean operation is cancelled.
Data parity or RAM error on data read	An error is flagged through the **PARRDINTR/ERRRDINTR** interrupt signals, the clean operation is cancelled.

Invalidate maintenance operation

Table 2-18 shows the Invalidate maintenance operation error signalling cases and effects.

Table 2-18 Invalidate maintenance operation cases

Case	Effect
Tag parity or RAM error on cache lookup, for invalidate by address operation	An error is flagged through the **PARRTINTR/ERRTRINTR** interrupt signals, the invalidate operation is cancelled.
Tag RAM error on valid information write	An error is flagged through the **ERRWTINTR** interrupt signal, subsequent reads to that line are unpredictable.

Clean and Invalidate maintenance operation

Table 2-19 shows the Clean and Invalidate maintenance operation error signalling cases and effects.

Table 2-19 Clean and Invalidate maintenance operation cases

Case	Effect
Tag parity or RAM error on cache lookup	An error is flagged through the **PARRTINTR/ERRRTINTR** interrupt signals, the clean operation is cancelled and the line is invalidated only for a clean and invalidate by the index operation, having no risk of unexpected hit.
Data parity or RAM error on data read	An error is flagged through the **PARRTINR/ERRRDINTR** interrupt signals, the clean operation is cancelled and the line is invalidated only for a clean and invalidate by index operation, having no risk of unexpected hit.
Tag RAM error on valid information write	An error is flagged through the **ERRWTINTR** interrupt signal, subsequent reads to that line are unpredictable.

L220 Cache Controller and system clocking

The default mode for Cache Controller is asynchronous mode, where the IEM register slices are used and the Cache Controller can be clocked with a different clock to the system. Synchronous mode can be achieved by asserting **SYNCMODEREQP**, **SYNCMODEREQM0** and **SYNCMODEREQM1** high for the peripheral, **M0 AXI** and **M1 AXI** ports.

The AXI master, slave and peripheral ports have their own clock enable to permit the transfer rate on the corresponding bus to be a submultiple of the Cache Controller clock.

2.2.8 Data RAM configuration

With lockdown format C, a block of the L2 cache can be used as a *Tightly Coupled Memory* (TCM). The effective size of the L2 cache can be from 16KB to 2MB, determined by the choice of the L2 cache size configuration of 128KB, 256KB, 512KB, 1MB or 2MB, and by the use of the lockdown register to reduce the associativity. Setting the ways in the Lockdown registers stops data in the cache from being evicted from the locked ways. Data can still be written to these ways if they hit. There are two lockdown registers, one for data and one for instructions.

Table 2-20 shows cache controller Data RAM sizes.

Table 2-20 Cache controller data RAM sizes

L2 Cache size configuration	Way size	L2 Effective write size							
Number of locked ways (available ways)		7 (1)	6 (2)	5 (3)	4 (4)	3 (5)	2 (6)	1 (7)	0 (8)
128KB	16KB	16KB	32KB	48KB	64KB	80KB	96KB	112KB	128KB
256KB	32KB	32KB	64KB	96KB	128KB	160KB	192KB	224KB	256KB
512KB	64KB	64KB	128KB	196KB	256KB	320KB	384KB	448KB	512KB
1MB	128KB	128KB	256KB	384KB	512KB	640KB	768KB	896KB	1MB
2MB	256KB	256KB	512KB	768KB	1MB	1.280MB	1.536MB	1.792MB	2MB

To simplify the design, and to ensure timing closure for all cache sizes, the tag width is 20 bits without parity or 21 bits with parity, defined by the smallest cache size, 128KB. When larger cache sizes are required, less of the bits are used to determine the Tag lookup. It is assumed in the following table that the data RAM width is one cache line, 256 bits, the tags are all read in parallel, and that the eight ways are stored continuously in the data RAM. Table 2-21 shows cache size and the cache controller RAM interface data.

Table 2-21 Cache size and cache controller RAM Interface data

L2 Cache Size	Way Size	32-bit address		L220 RAM Interface			
		Tag	Index, Word per Way	Data RAM width	Address bits per way	Data RAM Address	Tag RAM Address
128KB	16KB	31:14	13:2	255:0	8:0	11:0	8:0
256KB	32KB	31:15	14:2	255:0	9:0	12:0	9:0
512KB	64KB	31:16	15:2	255:0	10:0	13:0	10:0
1MB	128KB	31:17	16:2	255:0	11:0	14:0	11:0
2MB	256KB	31:18	17:2	255:0	12:0	15:0	12:0

2.2.9 RAM interfaces

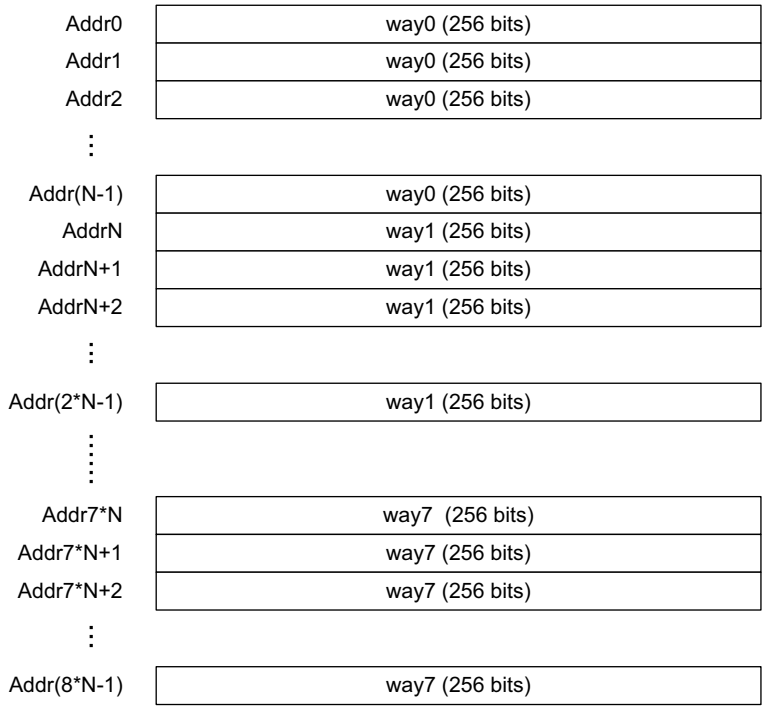
This section describes the following:

- *Data RAM*
- *Dirty RAM* on page 2-44
- *Tag RAM* on page 2-46
- *Cache lookup* on page 2-47
- *RAM sizes* on page 2-49
- *RAM bus usage versus cache associativity and way size* on page 2-49.

Data RAM

The data RAM shown in Figure 2-4 on page 2-43 is organized as 8 ways 256-bit wide contiguous memories. It supports the following accesses:

- 8 word data reads
- n * 8 bits data writes with byte enables controls
- 8 word data writes for linefills.



L2 Cache Size	N =
128KB	512
256KB	1,024
512KB	2,048
1MB	4,096
2MB	8,192

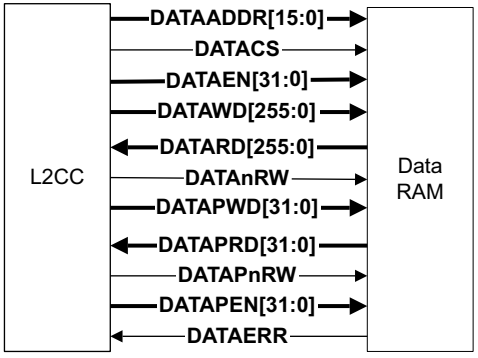


Figure 2-4 Data RAM organization

Dirty RAM

The dirty RAM shown in Figure 2-5 on page 2-45 is organized as a 16-bit wide memory, 2 bits per 8-word cache line. The dirty RAM address is the same as the tag RAM address bus. It supports the following accesses:

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

L2 Cache Size	N =
128KB	512
256KB	1,024
512KB	2,048
1MB	4,096
2MB	8,192

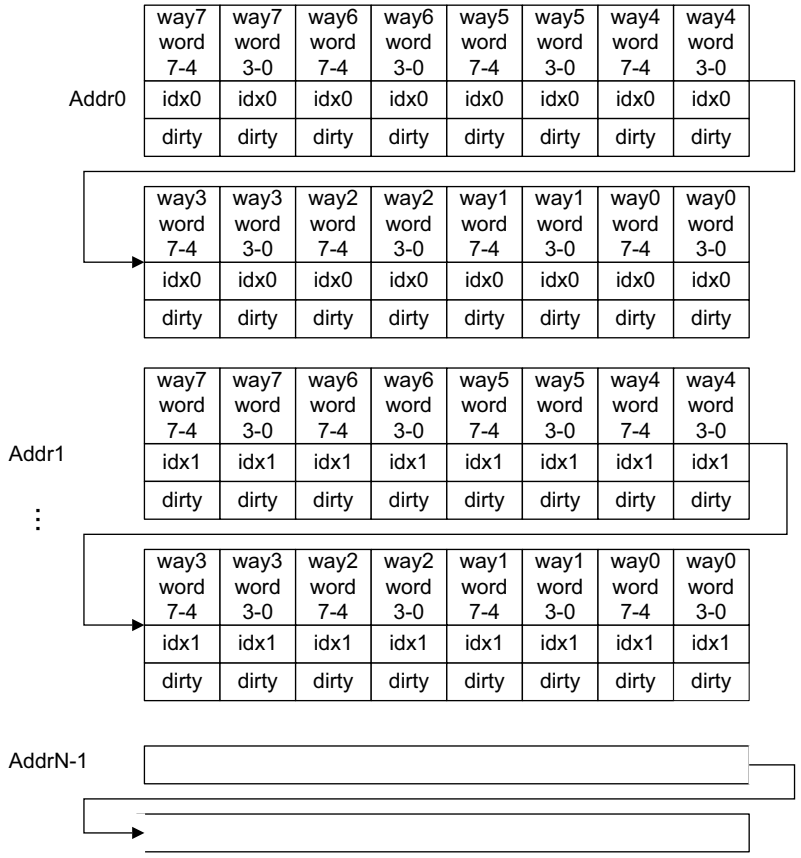


Figure 2-5 Dirty RAM organization

Figure 2-6 on page 2-46 shows the dirty RAM connectivity.

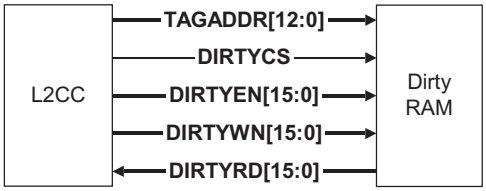


Figure 2-6 Dirty RAM connectivity

Tag RAM

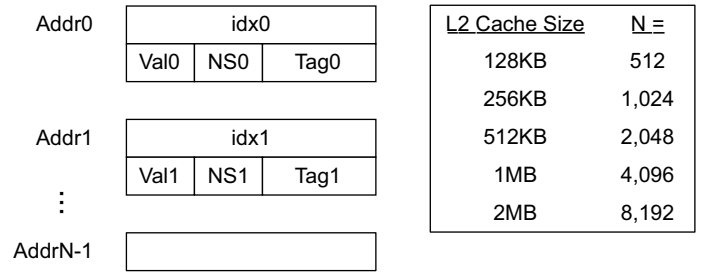
The tag RAM is shown in Figure 2-7 on page 2-47. There is one tag RAM for each way of the L2 cache. A tag RAM is organized as a 21-bit wide memory. 18 bits are dedicated to address tag, 1 bit for security information, 1 bit for valid information, and optionally 1 bit for parity. The tag RAM address bus is also the address bus for the dirty RAM. The tag RAM support the following accesses:

- 20-bit tag reads for Tag lookup
- 20-bit tag writes for allocations.

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

Note

You require a 21-bit wide memory to support the parity option.



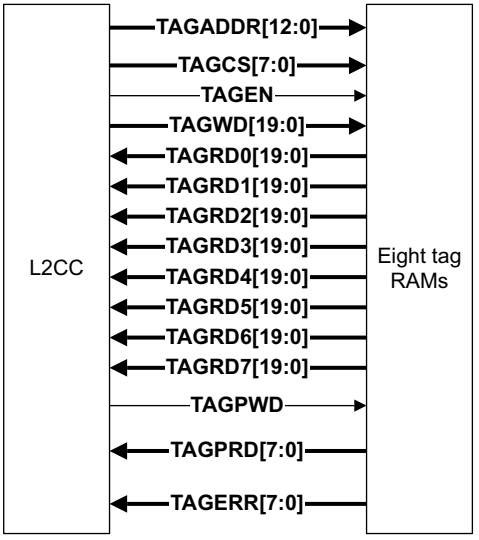


Figure 2-7 Tag RAM organization

Cache lookup

Figure 2-8 on page 2-48 shows the tag RAM format:

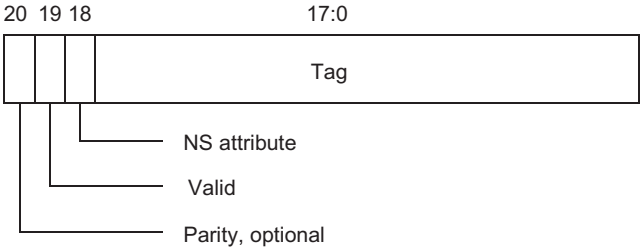


Figure 2-8 Tag RAM format

Each line is marked as secure or NS depending on the value of the **AWPROT[1]** or **ARPROT[1]** value on the original transaction. The security setting of the access, **AWPROT[1]** or **ARPROT[1]**, is used for Cache Lookup and compared with the NS attribute in the Tag.

The tag RAM contains a field to hold the NS attribute bit corresponding for each cache line. This is required so that the NS attribute bit for all cache ways is compared to generate the cache hit.

Note

- The cache is not automatically flushed when the processor changes security state.
- If an access is performed, and has an **AWPROT[1]/ARPROT[1]** value of 1'b1, then the NS attribute must be HIGH. Cache lookups are performed on lines marked as NS, the NS cache line attribute = 1, according to *Physical Address* (PA).
- If any access is performed in secure state, and the transaction has an **AWPROT[1]/ARPROT[1]** value of 1'b0), then the NS attribute must be LOW. Cache lookups are performed on lines marked as secure (NS cache line attribute = 0) according to PA. A secure access only hits on tags with a secure NS attribute.

RAM sizes

Table 2-22 shows the RAM sizes.

Table 2-22 RAM sizes

L2 cache size	Data RAM	Tag RAM	Dirty RAM
128KB	1 × (256 + 32) × (ways × 512)	Ways × (20 + 1) × 512	1 × (2 × ways) × 512
256KB	1 × (256 + 32) × (ways × 1024)	Ways × (19 + 1) × 1,024	1 × (2 × ways) × 1,024
512KB	1 × (256 + 32) × (ways × 2048)	Ways × (18 + 1) × 2,048	1 × (2 × ways) × 2,048
1MB	1 × (256 + 32) × (ways × 4096)	Ways × (17 + 1) × 4,096	1 × (2 × ways) × 4,096
2MB	1 × (256 + 32) × (ways × 8192)	Ways × (16 + 1) × 8,192	1x (2 × ways) × 8,192

Note

- The format for RAM sizes are:
  - Number of RAM × (width + parity) × number of address location
- The dirty ram does not have parity. Width for the tag RAM consists of Valid + NS + address.

RAM bus usage versus cache associativity and way size

This section covers:

- Data RAM usage
- Dirty RAM usage
- Tag RAM usage.

Data RAM usage

Figure 2-9 shows the **DATAADDR** bus format:

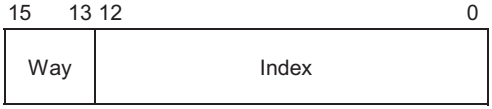


Figure 2-9 Data RAM address bus format

Bits [15:13] connections depend on the cache associativity:

- 1 or 2 way associative:
  - Bits [15:14] are left unconnected.

Bit 13 is the *Most Significant Bit* (MSB) of the RAM address bus.

- 3 or 4 way associative:
  - Bit 15 is left unconnected.

    Bits [14:13] are the MSB of the RAM address bus.
- 5 to 8 way associative: Bits [15:13] are the MSB of the RAM address bus.

Bits [12:0] depend on the way size:

- 16KB way size:
  - Bits [12:9] are left unconnected.

    Bits [8:0] are the *Least Significant Bits* (LSB) of the RAM address bus.
- 32KB way size:
  - Bits [12:10] are left unconnected.

    Bits [9:0] are the LSB of the RAM address bus.
- 64KB way size:
  - Bits [12:11] left unconnected.

    Bits [10:0] are the LSB of the RAM address bus.
- 128KB way size:
  - Bit 12 is left unconnected.

    Bits [11:0] are the LSB of the RAM address bus.
- 256KB way size:
  - Bits [12:0] are the LSB of the RAM address bus.

All bits of the data bus are always connected.

Dirty RAM usage

Depending on the way size, some bits of the **TAGADDR** bus are not used for the dirty RAM address:

- 16KB way size:
  - Bits [12:9] are left unconnected.

    Bits [8:0] are the LSB of the RAM address bus.
- 32KB way size:
  - Bits [12:10] are left unconnected.

    Bits [9:0] are the LSB of the RAM address bus.
- 64KB way size:
  - Bits [12:11] are left unconnected.

    Bits [10:0] are the LSB of the RAM address bus.

- 128KB way size:
  - Bit 12 is left unconnected.

    Bits [11:0] are the LSB of the RAM address bus.
- 256KB way size:
  - Bits [12:0] are the LSB of the RAM address bus.

Depending on the cache associativity, some bits of **DIRTYRD** bus must be tied low and some bits of **DIRTYWD** are left unconnected:

- 1 way associative:
  - Bits [1:0] of **DIRTYRD** and **DIRTYWD** used for data busses.

    Bits [15:2] of **DIRTYD** must be tied LOW.
- 2 way associative:
  - Bits [3:0] of **DIRTYRD** and **DIRTYWD** used for data busses.

    Bits [15:4] of **DIRTYD** must be tied LOW.
- 3 way associative:
  - Bits [5:0] of **DIRTYRD** and **DIRTYWD** used for data busses. Bits [15:6] of **DIRTYD** must be tied LOW.
- 4 way associative:
  - Bits [7:0] of **DIRTYRD** and **DIRTYWD** used for data buses.

    Bits [15:8] of **DIRTYD** must be tied LOW.
- 5 way associative:
  - Bits [9:0] of **DIRTYRD** and **DIRTYWD** used for data busses. Bits [15:10] of **DIRTYD** must be tied LOW.
- 6 way associative:
  - Bits [11:0] of **DIRTYRD** and **DIRTYWD** used for data busses.

    Bits [15:12] of **DIRTYD** must be tied LOW.
- 7 way associative:
  - Bits [13:0] of **DIRTYRD** and **DIRTYWD** used for data busses.

    Bits [15:2] of **DIRTYD** must be tied LOW.
- 8 way associative:
  - All bits [1:0] of **DIRTYRD** and **DIRTYWD** used for data busses.

Tag RAM usage

TAGADDR and **TAGRDn/TAGWDn** buses usage depends on the way size:

- 16KB way size:
  - Bits [12:9] of **TAGADDR** are left unconnected.

    Bits [8:0] are the RAM address bus. All bits of **TAGRDn** and **TAGWDn** are used for data busses.
- 32KB way size:
  - Bits [12:10] of **TAGADDR** are left unconnected. Bits [9:0] are the RAM address bus.

    Bits [19:1] of **TAGRDn** and **TAGWDn** are used for data buses.

    Bit 0 of **TAGRDn** must be tied LOW.
- 64KB way size:
  - Bits [12:11] of **TAGADDR** are left unconnected.

    Bits [10:0] are the RAM address bus. All bits of **TAGRDn** and **TAGWDn** are used for data buses.

    Bits [1:0] of **TAGRDn** must be tied LOW.
- 128KB way size:
  - Bit 12 of **TAGADDR** is left unconnected.

    Bits [11:0] are the RAM address bus.

    Bits [19:4] of **TAGRDn** and **TAGWDn** are used for data busses.

    Bits [2:0] of **TAGRDn** must be tied LOW.
- 256KB way size:
  - Bits [12:0] are the RAM address bus.

    Bits [19:4] of **TAGRDn** and **TAGWDn** are used for data busses.

    Bits [3:0] of **TAGRDn** must be tied LOW.

Chapter 3
Programmer's Model

This chapter describes the L220 Cache Controller registers and provides information for programming the device. It contains the following sections:

3.1 About the programmer's 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.
- Reserved or unused address locations must not be accessed because this can result in unpredictable behavior of the device.
- All registers support read and write accesses unless otherwise stated in the relevant text. A write updates the contents of a register and a read returns the contents of the register.
- All cache maintenance operations automatically perform a Cache Sync operation before starting. Before writing to any other register you must perform an explicit Cache Sync operation. This is particularly important when the cache is enabled and changes to how the cache allocates new lines are to be made.

3.1.1 Configuration and control registers

The cache controller is controlled through a set of memory-mapped registers that occupy a re-locatable 4KB of memory. You can program the registers using the 32-bit AXI peripheral port. **AWADDRP [11:0]** or **ARADDRP[11:0]** are decoded to select the register being accessed. The **AWPROTP [1]** or **ARPROTP[1]** value is checked to see if the transaction is permitted to access the register.

Warning

At boot time you must perform a Secure write to the Invalidate by Way/Index, `0x77C` Register 7, 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 Auxiliary Control Register using a read-modify-write to set up global configurations:
   - Associativity, Way Size
   - Latencies for RAM accesses.
2. Secure write to the Invalidate by Way/Index, `0x77C` Register 7, to invalidate all entries in cache:
   - Write of `FF` to `77C`
   - 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 Control Register 1 with the LSB set to 1 to enable the cache.
6. Finally poll the Control Register until the control sequence is complete.

If you write to the Auxiliary 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 any of the Auxiliary Control Registers. You can program all the registers using the AXI peripheral interface.

Note

All unmapped registers are read as zero. All unused bits in mapped registers are also read as zero.

3.2 Summary of registers

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

Table 3-1 Cache controller register map

Register	Reads	Writes	Secure
0	System ID and Cache Type	Ignored	NS
1	Control	Control	Write S Read NS/S
2	Interrupt/Counter Control	Interrupt/Counter Control	NS
3-6	Reserved	Reserved	-
7	Cache Maintenance Operations	Cache Maintenance Operations	Secure bit of access effects operation
8	Reserved	Reserved	-
9	Cache Lockdown	Cache Lockdown	Secure bit of access effects operation
10-14	Reserved	Reserved	-
15	Test and Debug	Test and Debug	Write S Read 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

Register	Name	Base offset	Type	Reset value	Description
r0	Cache ID	0x000	RO	0x41000086[a]	*Register 0,Cache ID Register* on page 3-7
r0	Cache Type	0x004	RO	0x1C100100	*Register 0,Cache Type Register* on page 3-8
r1	Control	0x100	RW	0x00000000	*Register 1,Control Register* on page 3-9
r1	Auxiliary Control	0x104	RW	0x02020FFF	*Register 1, Auxiliary Control Register* on page 3-10
r2	Event Counter Control	0x200	RW	0x00000000	*Register 2, Event Counter Control Register* on page 3-13 on page 3-13
r2	Event Counter1 Configuration	0x204	RW	0x00000000	*Register 2, Event Counter0 Configuration Register* on page 3-15

Table 3-2 Summary of Cache Controller registers (continued)

Register	Name	Base offset	Type	Reset value	Description
r2	Event Counter0 Configuration	0x208	RW	0x00000000	*Register 2, Event Counter0 Configuration Register* on page 3-15
r2	Event Counter1 Value	0x20C	RW	0x00000000	*Register 2, Event Counter1 Value Register* on page 3-17
r2	Event Counter0 Value	0x210	RW	0x00000000	*Register 2, Event Counter1 Value Register* on page 3-17
r2	Interrupt Mask[b]	0x214	RW	0x00000000	*Register 2, Interrupt Mask Register* on page 3-17
r2	Masked Interrupt Status[b]	0x218	RO	0x00000000	*Register 2, Masked Interrupt Status Register* on page 3-19
r2	Raw Interrupt Status[b]	0x21C	RO	0x00000000	*Register 2, Raw Interrupt Status Register* on page 3-20
r2	Interrupt Clear[b]	0x220	WO	0x00000000	*Register 2, Interrupt Clear Register* on page 3-21
r7	Cache Sync	0x730	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Invalidate Line By PA	0x770	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Invalidate by Way	0x77C	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Clean Line by PA	0x7B0	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Clean Line by Index/Way	0x7B8	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Clean by Way	0x7BC	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Clean and Invalidate Line by PA	0x7F0	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r7	Clean and Invalidate Line by Index/Way	0x7F8	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22

Table 3-2 Summary of Cache Controller registers (continued)

Register	Name	Base offset	Type	Reset value	Description
r7	Clean and Invalidate by Way	0x7FC	RW	0x00000000	*Register 7, Cache Maintenance Operations* on page 3-22
r9	Lockdown by Way - D Side	0x900	RW	0x00000000	*Register 9, Cache Lockdown* on page 3-27
r9	Lockdown by Way - I Side	0x904	RW	0x00000000	*Register 9, Cache Lockdown* on page 3-27
r15	Test Operation	0xF00	RW	0x00000000	*Register 15, Test and Debug* on page 3-31
r15	Line Data (8 × Word)	0xF10, 0xF14, 0xF18, 0xF1C, 0xF20, 0xF24, 0xF28, and 0xF2C	RW	Not reset	*Register 15, Test and Debug* on page 3-31
r15	Line Tag {Tag, V, D0, D1, RR}	0xF30	RW	Not reset	*Register 15, Test and Debug* on page 3-31
r15	Debug Control Register	0xF40	RW	0x00000000	*Register 15, Test and Debug* on page 3-31

a. This value is L220 pin dependent, depending on how external CACHEID pins are tied.
b. The cache interrupt registers are those that can be accessed by secure and NS operations.

3.3 Register descriptions

This section describes the cache controller registers. Table 3-2 on page 3-4 shows cross references to individual registers.

3.3.1 Register 0,Cache ID Register

This is a read-only register and returns the 32-bit device ID code. This register reads off the **CACHEID** input bus, the value specified by system integrator.

Figure 3-1 shows the register bit assignments.

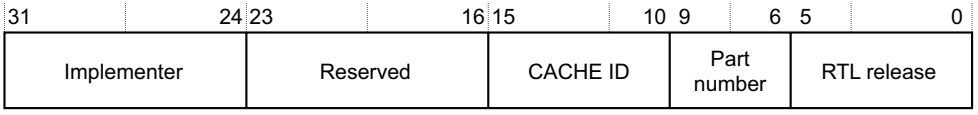


Figure 3-1 Register 0, Cache ID bit assignments

Table 3-3 shows the register bit assignments.

Table 3-3 Register 0,Cache ID bit assignments

Bits	Name	Description
[31:24]	Implementor	0x41 (ARM)
[23:16]	Reserved	SBZ
[15:10]	CACHE ID	-
[9:6]	Part number	0x2
[5:0]	RTL release	0x6

Note

- Part number 0x2 denotes ARM L220 Level 2 Cache Controller
- RTL release 0x6 denotes r1p7-01rel0 RTL code of the Cache Controller. Refer to the Release Note accompanying the release of the L220 Cache Controller to find out the value of these bits for other releases

3.3.2 Register 0,Cache Type Register

The Cache Type Register is a read-only register and 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.

Figure 3-2 shows the register bit assignments.

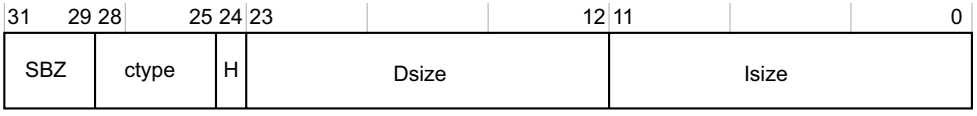


Figure 3-2 Register 0,Cache Type bit assignments

Table 3-4 shows the register bit assignments.

Table 3-4 Register 0,Cache Type bit assignments

Bits	Field	Sub-field	Comments
[31:29]	SBZ		3'b000
[28:25]	ctype		4'b1110 - Lockdown format C
[24]	H		1'b0 - unified[a]
[23:12]	Dsize		Usize
	[23]	SBZ / RAZ	1'b0
	[22:20]	L2 cache way size	Read from Auxiliary Control Register [19:17]
	[19]	SBZ / RAZ	1'b0
	[18:15]	L2 associativity	Read from Auxiliary Control Register [16:13]
	[14]	SBZ	1'b0
	[13:12]	L2 cache line length	2'b00 - 32 bytes
[11:0]	Isize		Usize
	[11]	SBZ / RAZ	1'b0
	[10:8]	L2 cache way size	Read from Auxiliary Control Register [19:17]
	[7]	SBZ / RAZ	1'b0

Table 3-4 Register 0,Cache Type bit assignments (continued)

Bits	Field	Sub-field	Comments
	[6:3]	L2 associativity	Read from Auxiliary Control Register [16:13]
	[2]	SBZ	1'b0
	[1:0]	L2 cache line length	2'b00 - 32 bytes

a. 1'b1 = Harvard.

3.3.3 Register 1,Control Register

The Control Register is a read and write register. This register enables or disables the cache controller and consequently all transactions are passed through to L3 memory. If disabled, the cache controller is placed in bypass mode. This register 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 outstanding transactions to complete.
2. Return a write response and perform a cache sync operation in the background.
3. Update register when a cache sync operation is complete.

Figure 3-3 shows the register bit assignments.

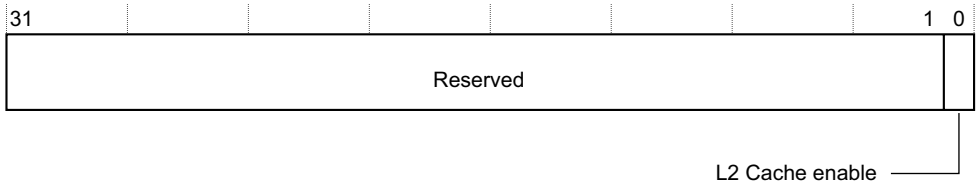


Figure 3-3 Register 1,Control bit assignments

Table 3-5 shows the register bit assignments.

Table 3-5 Register 1,Control bit assignments

Bit	Field	Description
[31:1]	Reserved	SBZ / RAZ
[0]	L2 Cache enable	0 = L2 Cache is in bypass mode, this is the default value. 1 = L2 Cache is enabled.

3.3.4 Register 1, Auxiliary Control Register

The Auxiliary Control Register is a read and write register. This register enables you to configure:

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

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

Figure 3-4 shows the register bit assignments.

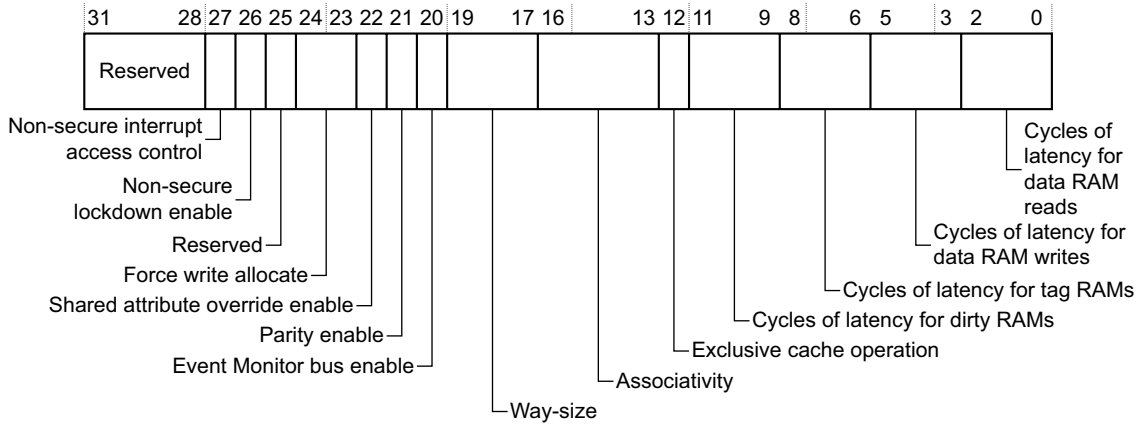


Figure 3-4 Auxiliary Control Register

Table 3-6 shows the register bit assignments.

Table 3-6 Auxiliary Control Register

Bit	Field	Description
[31:28]	Reserved	SBZ / RAZ
[27]	Non-secure interrupt access control	0 = Interrupt Clear (`0x220`) and Interrupt Mask (`0x214`) can only be modified or read with secure accesses, the default value 1 = Interrupt Clear (`0x220`) and Interrupt Mask (`0x214`) can be modified or read with secure accesses or non-secure accesses
[26]	Non-secure lockdown enable	0 = Lockdown registers cannot be modified using NS accesses, this is the default 1 = NS accesses can write to the lockdown registers
[25]	Reserved	SBO / RAO
[24:23]	Force write allocate	2'b00 = Use **AWCACHE** attributes for WA, this is the default 2'b01 = Force no allocate, set WA bit always `0` 2'b10 = Override **AWCACHE** attributes, set WA bit always 1 (all cacheable write misses become write allocated) 2'b11 = internally mapped to 2'b00 See *Behavior for ARMv6 transactions* on page 2-12 for more details
[22]	Shared attribute override enable	0 = Shared accesses treated as Non-Cacheable (NC), this is the default value. 1 = Shared attribute internally ignored.
[21]	Parity enable	0 = Disabled, this is the default value 1 = Enabled
[20]	Event monitor bus enable	Event Monitor bus enable
[19:17]	Way-size	3'b000 = Reserved, internally mapped to 16KB 3'b001 = 16KB, this is the default value 3'b010 = 32KB 3'b011 = 64KB 3'b100 = 128KB 3'b101 = 256KB 3'b110 to 3'b111 = Reserved, internally mapped to 256 KB

Table 3-6 Auxiliary Control Register (continued)

Bit	Field	Description
[16:13]	Associativity	4'b0000 = cache absent, this is the default value. 4'b0001 = direct-mapped cache 4'b0010 = 2-way 4'b0011 = 3-way 4'b0100 = 4-way 4'b0101 = 5-way 4'b0110 = 6-way 4'b0111 = 7-way 4'b1000 = 8-way 4'b1001 to 4'b1111 = Reserved, internally mapped to 8-way
[12]	Exclusive cache operation	0 = Disabled, this is the default 1 = Enabled, see *Exclusive cache configuration* on page 2-15.
[11:9]	Cycles of latency for dirty RAM	3'b000 = 1 cycle of latency, there is no additional latency 3'b001 = 2 cycles of latency 3'b010 = 3 cycles of latency 3'b011 = 4 cycles of latency 3'b100 = 5 cycles of latency 3'b101 = 6 cycles of latency 3'b110 = 7 cycles of latency 3'b111 = 8 cycles of latency, the default value
[8:6]	Cycles of latency for tag RAMs	
[5:3]	Cycles of latency for data RAM writes	
[2:0]	Cycles of latency for data RAM reads	

Note

When a shared attribute override bit is set, the shared attribute **AWSIDEBANDSx[0]/ARSIDEBANDSx[0]** is internally ignored. For noncacheable, nonbufferable and linefill transfers transactions, these are forwarded to L3.

L220 Cache Controller Compiled RAM latencies

The Cache Controller resets assuming the slowest compiled RAMs are being used. This means eight Cache Controller clock cycles are used for each access. In terms of reads, this means that the read data is sampled eight clock edges after the edge on which the RAM samples the read request. Using this arrangement, the shortest latency is one. You can program the latencies for each RAM in the Auxiliary Control Register. You must only program this register when the L2 cache is not enabled. Figure 3-5 on page 3-13 shows a compiled RAM latency.

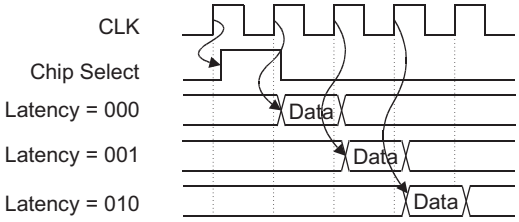


Figure 3-5 Compiled RAM Latency

In the following circumstances there are some dependencies on how the RAM latencies are used, when it is required to access two or more of the cache memories in parallel.

Line TAG lookups:

If the tag RAM latency is larger than the dirty RAM latency, the tag memory latency is used, otherwise all write-back accesses use the dirty RAM latency.

Allocations:

If the data RAM write or dirty RAM latency is larger than the tag memory latency, the larger latency configuration of the two, either data or dirty RAM latency is used, otherwise all allocations use the tag RAM latency.

3.3.5 Register 2, Event Counter Control Register

The Event Counter Control Register is a read and write register. This register permits the event counters to be enabled and reset when required. This register can be accessed by secure and NS operations. Figure 3-6 shows the register bit assignments.

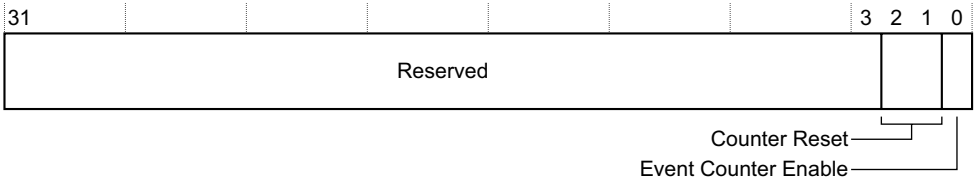


Figure 3-6 Event Counter Control Register bit assignments

Table 3-7 shows the register bit assignments.

Table 3-7 Event Counter Control Register bit assignments

Bits	Field	Description
[31:3]	Reserved	SBZ / RAZ
[2:1]	Counter Reset	Always Read as zero Corresponding counters are reset when the bit is written to by 1'b1: Bit 1 = Event Counter1 reset Bit 0 = Event Counter0 reset
[0]	Event Counter Enable	0 = Event Counting Disable, the default value 1 = Event Counting Enable

3.3.6 Register 2, Event Counter1 Configuration Register

The Event Counter1 Configuration Register is a read and write register. This register enables the event counter1 to be driven by a specific event. When the event occurs it causes the counter1 to increment. The counter event source signals are described in *Cache event monitoring* on page 2-34. This register can be accessed by secure and NS operations. Figure 3-7 shows the register bit assignments.

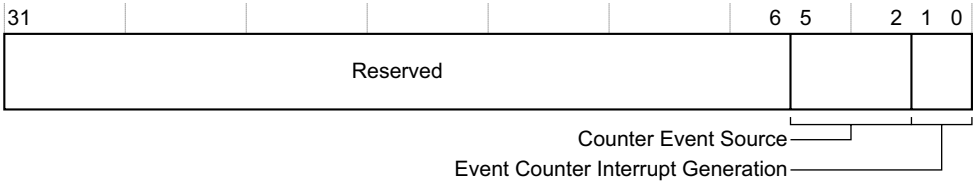


Figure 3-7 Event Counter1 Configuration Register bit assignments

Table 3-8 shows the register bit assignments.

Table 3-8 Event Counter1 Configuration Register bit assignments

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

Table 3-8 Event Counter1 Configuration Register bit assignments (continued)

Bits	Field	Description	
[5:2]	Counter Event Source	Event	Encoding
		Counter Disabled	b0000
		CO	b0001
		DRHIT	b0010
		DRREQ	b0011
		DWHIT	b0100
		DWREQ	b0101
		DWTREQ	b0110
		IRHIT	b0111
		IRREQ	b1000
		WA	b1001
		Counter Disabled	b1010-1111
[1:0]	Event Counter Interrupt Generation	00 = Disabled (default)	
		01 = Enabled: Increment condition	
		10 = Enabled: Overflow condition	
		11 = Interrupt Generation is disabled	

Note

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

3.3.7 Register 2, Event Counter0 Configuration Register

The Event Counter0 Configuration Register is a read and write register. This register enables the event counter0 to be driven by a specific event, when the event occurs it causes the counter0 to increment. This register can be accessed by secure and NS operations. The counter event source signals are described in *Cache event monitoring* on page 2-34. Figure 3-8 on page 3-16 shows the register bit assignments.

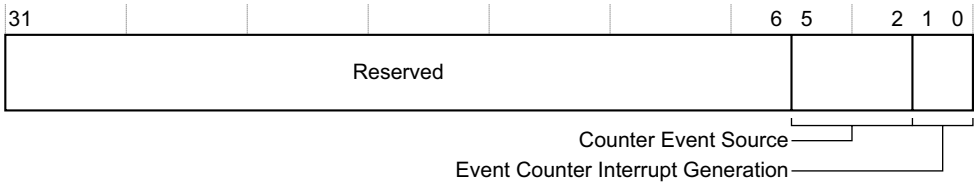


Figure 3-8 Event Counter0 Configuration Register bit assignments

Table 3-9 shows the register bit assignments.

Table 3-9 Event Counter0 Configuration Register bit assignments

Bits	Field	Description	
[31:6]	Reserved	SBZ / RAZ	
[5:2]	Counter Event Source	Event	Encoding
		Counter Disabled	b0000
		CO	b0001
		DRHIT	b0010
		DRREQ	b0011
		DWHIT	b0100
		DWREQ	b0101
		DWTREQ	b0110
		IRHIT	b0111
		IRREQ	b1000
		WA	b1001
		Counter Disabled	b1010-1111
[1:0]	Event Counter Interrupt Generation	00 = Disabled (default)	
		01 = Enabled: Increment condition	
		10 = Enabled: Overflow condition	
		11 = Interrupt Generation is disabled	

Note

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

3.3.8 Register 2, Event Counter1 Value Register

The Event Counter1 Value Register is a read and write register. This register enables the programmer to read off the counter value. The counter counts an event as specified by the counter1 configuration register. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register. This register can be accessed by secure and NS operations. Table 3-10 shows the register bit assignments. This register can only be written to when bits [5:2] of the L2 Event Counter1 Configuration Register are set to Counter Disabled.

Table 3-10 Event Counter 1 Value Register bit assignments

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

3.3.9 Register 2, Event Counter0 Value Register

The Event Counter0 Value Register is a read and write register. This register enables the programmer to read off the counter value. The counter counts an event as specified by the Counter0 Configuration Register. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register. This register can be accessed by secure and NS operations. Table 3-11 shows the register bit assignments. This register can only be written to when bits [5:2] of the L2 Event Counter0 Configuration Register are set to Counter Disabled.

Table 3-11 Event Counter 0 Value Register bit assignments

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

3.3.10 Register 2, Interrupt Mask Register

The Interrupt Mask Register is a read and write register. This register enables or masks interrupts from being triggered on the external pins of the Cache Controller. This register can be accessed by secure and NS operations. Figure 3-9 on page 3-18 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. The reset value for this register is 32'h00000000. NS access to this register is dependent on Auxiliary Control Register bit [27].

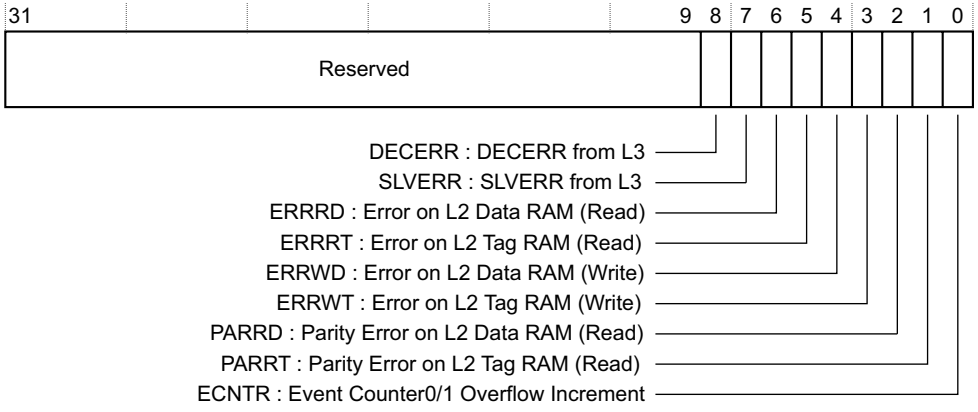


Figure 3-9 Interrupt Mask Register bit assignments

Table 3-12 shows the register bit assignments.

Table 3-12 Interrupt Mask Register bit assignments

Bits	Field	Description
[31:9]	Reserved	SBZ / RAZ
[8]	DECERR: DECERR from L3	1 = Enable 0 = Masked, this is the default
[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 Counter0/1 Overflow Increment	

3.3.11 Register 2, Masked Interrupt Status Register

The Masked Interrupt Status Register is a read-only register. This register is a read-only that enables the interrupt status that includes the masking logic. Figure 3-10 shows the register bit assignments. This register can be accessed by secure and NS operations. The register gives an AND function of the raw interrupt status with the values of the interrupt mask register. If bits read HIGH in this register they reflect the status of the input lines triggering an interrupt. If bits read LOW then it indicates that either no interrupt has been generated or the interrupt is masked. All the bits are cleared by a reset.

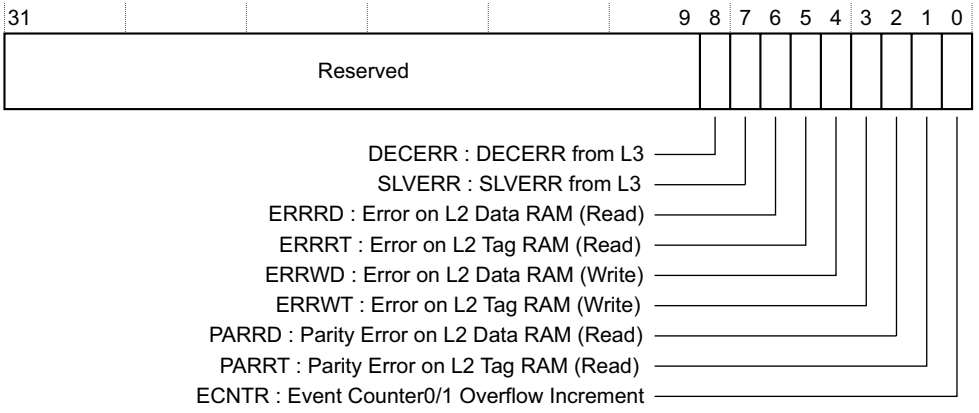


Figure 3-10 Masked Interrupt Status Register bit assignments

Table 3-13 shows the register bit assignments.

Table 3-13 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
[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)	

Table 3-13 Masked Interrupt Status Register bit assignments (continued)

Bits	Field	Description
[2]	PARRD: Parity Error on L2 data RAM (Read)	
[1]	PARRT: Parity Error on L2 tag RAM (Read)	
[0]	ECNTR: Event Counter0/1 Overflow Increment	

3.3.12 Register 2, Raw Interrupt Status Register

The Raw Interrupt Status Register is a read-only register that enables the interrupt status that excludes the masking logic. This register can be accessed by secure and NS operations. If bits read HIGH in this register they reflect the status of the input lines triggering an interrupt. If bits read LOW then it indicates that either no interrupt has been generated. All the bits are cleared by a reset. Figure 3-11 shows the register bit assignments.

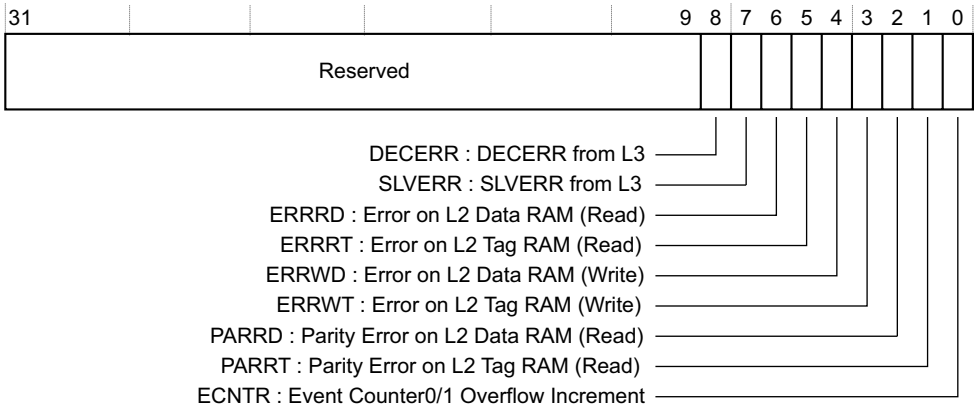


Figure 3-11 Raw Interrupt Status Register bit assignments

Table 3-14 shows the register bit assignments.

Table 3-14 Raw Interrupt Status Register bit assignments

Bits	Field	Description
[31:9]	Reserved	RAZ

Table 3-14 Raw Interrupt Status Register bit assignments (continued)

Bits	Field	Description
[8]	DECERR: DECERR from L3	Bits read can be HIGH or LOW
[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 Counter0/1 Overflow Increment	

3.3.13 Register 2, Interrupt Clear Register

The Interrupt Clear Register is a write-only register and all bits are cleared by the reset. Writing a 1 to a bit in this register clears the corresponding bit in the Raw Interrupt Status Register. Writing a 0 has no effect. NS access to this register is dependent on Auxiliary Control Register bit [27]. Figure 3-12 shows the register bit assignments.

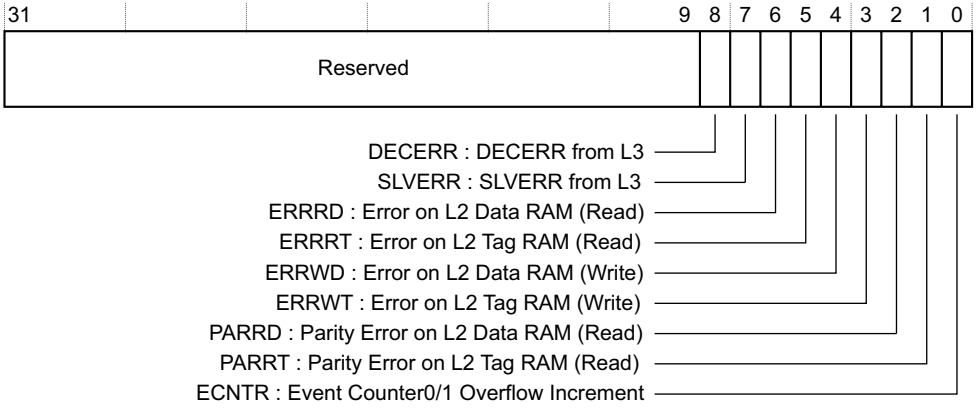


Figure 3-12 Interrupt Clear Register bit assignments

Table 3-15 shows the register bit assignments.

Table 3-15 Interrupt 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 Counter0/1 Overflow Increment	

3.3.14 Register 7, Cache Maintenance Operations

The Cache Maintenance Operations registers have different behavior, depending on the AXI security flag of the access requesting a cache operation. If the operation is specific to the Way or Index/Way then 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 NS lines.

Note

To invalidate the whole cache at boot time, you must use a secure access to the Invalidate by Way operation.

Non-secure access

The secure bit of the tag is checked, a lookup must be done for each NS maintenance operation, and the maintenance operation can only affect NS 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 PA then the behavior is presented in the following manner:

- Secure access: The data in the cache is only affected if by the operation if it is secure.
- Non-secure access: The data in the cache is only affected if by the operation if it is NS.

Table 3-16 shows the cache maintenance operations. They are executed by writing to the Cache Operations Register 7. See also Table 3-17 on page 3-26.

Table 3-16 Maintenance operations

Operation	Base offset	Type	Bit assignment format
Cache Sync	0x730	RW	See Figure 3-13 on page 3-24
Invalidate Line by PA	0x770	RW	See Figure 3-14 on page 3-24
Invalidate by Way	0x77C	RW	See Figure 3-16 on page 3-25
Clean Line by PA	0x7B0	RW	See Figure 3-14 on page 3-24
Clean Line by Index/Way	0x7B8	RW	See Figure 3-15 on page 3-25
Clean by Way	0x7BC	RW	See Figure 3-16 on page 3-25
Clean and Invalidate Line by PA	0x7F0	RW	See Figure 3-14 on page 3-24
Clean and Invalidate Line by Index/Way	0x7F8	RW	See Figure 3-15 on page 3-25
Clean and Invalidate by Way	0x7FC	RW	See Figure 3-16 on page 3-25

Background line operations

The following are background line operations:

- Clean Line by PA or by Index/Way
- Invalidate Line by PA
- Clean and Invalidate Line by PA or by Index/Way
- Cache Synchronization, Drain Linefill and WBs.

Writing to the register starts the operation, on the line specified by either {Tag, Index} or {Way, Index}. Tag and Index fields sizes depend on cache way size. When background operations are in progress the peripheral port generates a slave error if performing more transactions. The peripheral port is designed not to stall during background operations.

If a background task is requested by a write to the cache maintenance registers, while another background task is in progress and not completed, it is ignored and returns a SLVERR response.

For all background line and background way operations the C bit is set to 1, and it is reset to 0 when the operation has completed, so you must poll Register 7 to determine if there are any running background operations. The C bit can be read from several registers. See Figure 3-13, Figure 3-14, and Figure 3-15 on page 3-25 for the different formats.

Writing to the Cache Sync address of any data value initiates a synchronization of the L2 with main memory. It drains the data from the WB and EB to L3 memory and also drains the linefill and WABs to the cache data RAM.

Note

The L2 Cache Sync maintenance operation has the same behavior if accessed by a secure or NS transaction to the peripheral port.

For cache maintenance operations on a line, the line can be accessed through a PA or index way combination.

Figure 3-13 shows the Cache Sync Register. For cache sync operations, you must poll the register bit C for completion of the cache sync operation.

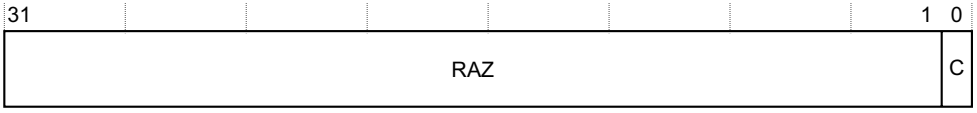


Figure 3-13 Cache Sync Register assignment

Figure 3-14 shows the PA format. For line based operations, the PA is given on **WDATAP[31:0]**.

Figure 3-14 Physical address format

Figure 3-15 on page 3-25 shows the Index or Way format. For line based operations, the Index or Way is given on **WDATAP[31:0]**.

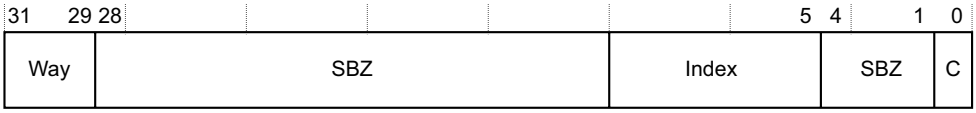


Figure 3-15 Index way combination format

Note

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

Background Way operations

The following operations are run as background tasks:

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

Writing to the register starts the operation on the Ways set to 1 in bits [7:0]. The Way bits are set to 1, and they are reset to 0 as their respective Ways are cleaned and invalidated. The C bit is set to 1, and it is reset to 0 when the operation has completed. See Figure 3-13 on page 3-24, Figure 3-14 on page 3-24, and Figure 3-15 for the formats of the different registers where the C bit is present. You must poll Register 7 to see when the operation is complete.

For way based operations, the way is given on **WDATAP[31:0]**, using the lockdown format C format. Figure 3-16 shows the Format C lockdown. You can select multiple ways at the same time, by setting the Way bits to 1.

Figure 3-16 Format C lockdown

During background operations, the software must not access these ways. If an atomic or a background task is requested by a write to the cache maintenance registers while a background task has not completed, it is ignored and returns a SLVERR response.

During background operations, the targeted way is considered to be locked, this means that no allocation occurs to that way on read or write misses. Read or write hits, are permitted to access the way. No hazard detection is performed on data returned,

additionally the data written might not be coherent with L3. This is because it is unknown whether the background operation has completed. In summary, there can still be dirty lines after a cache clean operation.

Note

Data accessed by the L1 master is still correct.

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

Note

Background operations have no effect when associativity is set to cache absent, that is, the Auxiliary Control Register bits [16:13] are set to 4'b0000.

Table 3-17 shows the cache maintenance operations.

Table 3-17 Cache maintenance operations

Operation	Description
Cache Sync	Drain the WB and EB to L3 main memory. Drain both linefill buffers (LFB0, LFB1) and WAB to the L2 data RAM.
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. Completes as a background task. An Invalidate by way while selecting all cache ways is equivalent to invalidating all cache entries. Completes as a background task with the way(s) locked, preventing allocation.
Clean Line by PA	Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.
Clean Line by Index/Way	Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.
Clean by Way	Writes each line of the specified L2 cache ways to L3 main memory if the line is marked as valid and dirty. The lines are marked as not dirty. The valid bits are unchanged. Completes as a background task with the way(s) locked, preventing allocation.

Table 3-17 Cache maintenance operations (continued)

Operation	Description
Clean and Invalidate Line by PA	Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid.
Clean and Invalidate Line by Index/Way	Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid.
Clean and Invalidate by Way	Writes each line of the specified L2 cache ways to L3 main memory if the line is marked as valid and dirty. The lines are marked as not valid. Completes as a background task with the way(s) locked, preventing allocation.

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

3.3.15 Register 9, Cache Lockdown

The Cache Lockdown Register is a read-write register. This register prevents new addresses being allocated and also prevents the data in the set ways from being evicted. It also enables the cache controller to filter data from instructions or data transactions. Cache lockdown is controlled by the Cache Lockdown Register 9. It is implemented as two lockdown subregisters, one for data and one for instructions. These registers have read-only or read write permission, depending on the security you have selected for the register access and on the Non-Secure Lockdown Enable bit in the Auxiliary Control Register. Table 3-18 shows the different settings of the Cache Lockdown Register.

Table 3-18 Cache lockdown

Security of the register access	Non-Secure Lockdown Enable bit	Permission
Secure	0 the default value, or 1	Read and Write
Non-Secure	0 the default value, or 1	Read only 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 NS accesses. In that configuration, if a NS 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 NS lockdown functionality can be identical to the secure one.

To only use selected cache Ways within a SET, Lockdown format C, defined by the *ARM Architecture Reference Manual* provides a method to restrict the replacement algorithm used on cache linefills, read this section for more information. Using this method, you can fetch code or load data into the L2 cache and protect it from being

evicted. Alternatively the method can be used to reduce cache pollution. See Figure 3-16 on page 3-25. The 32-bit ADDR cache address consists of the following fields: < TAG > < INDEX > < WORD > < BYTE >

Whenever the cache lookup occurs, the Index defines where in the cache ways to look, and the number of ways defines the number of locations with the same Index. This is called a Set. Therefore an 8-way set associative cache has eight locations where an address with INDEX (A) can exist.

Note

The number of locations are also known as lines.

If the cache lookup misses and a cache linefill is required, there are eight possible locations where the new line can be placed. Lockdown format C restricts the cache replacement algorithm to only use a subset of the eight possible locations. The locking configurability is listed in Table 3-19 and Table 3-20. To apply lockdown, set to 1'b1 for each bit to lock each respective Way. For example, set Bit [0] for Way 0, Bit [1] for Way 1.

Table 3-19 Data Lockdown Register - offset 0x900

Bit	Field	Description
[31:8]	Reserved	SBZ / RAZ
[7:0]	DATALOCK	Set to 1'b1 for each bit to lock each respective Way.

Table 3-20 Instruction Lockdown Register - offset 0x904

Bit	Field	Description
[31:8]	Reserved	SBZ / RAZ
[7:0]	INSTRLOCK	Set to 1b1 for each bit to lock each respective Way.

Fetching code or loading data into the L2 cache

Use lockdown format C in the following cases:

1. To make sure the selected way is unlocked, for example, unlock Way 0.
2. Loading code or data into the cache on Way 0.

3. Writing to the lockdown register to prevent filling to Way 0, but enable filling to Ways 1-7. The code/data is now protected in the cache and is not evicted on allocation.

Preventing and/or reducing cache pollution

There are benefits of having a critical piece of software or data being cached in the Cache Controller with out it being polluted or evicted, because of a new allocation. Using lockdown format C provides a simple method to load data into the Cache Controller through the linefill mechanism, then locking the cache memory to prevent eviction, and finally using the L220 cache maintenance operations to efficiently clean the data to the main memory system.

To do that:

1. Use lockdown format C, I and/or D lockdown, to restrict the permitted ways for cache filling to `n`-ways, where `n` is less than the total number of ways. For example, only permit filling to way 0.

2. Cache code or data into the cache
   - Fetch code into the cache by executing the routine for the first time
   - Load data into the cache by:
     — loading data for the first time
     — cache the data in L2 by executing the read loop being cached at L1 only

3. Write to the Lockdown Register, I and/or D lockdown, to prevent allocation to WAY 0, but to enable allocation to WAYS 1-7. The code/data is now protected in the cache and is not evicted on a linefill.

Example using a lockdown format C for the cache controller

There can be benefits in processing large frame buffers in the cache controller, and making them appear as if there is a large amount of restricted physically addressed space available in fast memory. Because the cache controller is 8-way set associative, using lockdown format C provides a simple method to load data into the cache controller using the linefill mechanism, lock the cache memory to prevent eviction, then use the cache controller cache maintenance operations to efficiently clean the data to the main memory system.

For example, if you require a 1MB Frame Buffer in the 2MB 8-way set associative cache controller, in a system using the ARM1176JZ(F)-S processor:

1. Set the address page attributes so the L1 is noncacheable and the L2 page is cacheable.

2. Set the L220 lockdown to only fill to Ways 0-3, this is equivalent to 1MB.
3. The 1MB is now contiguously mapped over 4 Ways of 256k each.
4. Load data into the L220 cache by executing a load routine in the ARM processor, where a series of LDRs are issued, a cache line apart from one another. The L1 attribute is noncacheable, so the L1 cache is not polluted. The L2 cache attribute is cacheable, so the Cache Controller performs linefills, filling into Ways 0-3. The linefill buffer provides efficient filling.
5. When finished, set the L220 lockdown to lock Ways 0-3, and only permit filling to Ways 4-7, this is equivalent to 1MB.

The L2 cache now contains a 1MB frame buffer, and 1MB of 4-way associative cache. Lookups and reads/writes can occur to the entire 2MB, but the frame buffer is prevented from being evicted. L220 cache maintenance operations can be used to efficiently clean to main memory. When the frame buffer is no longer required, Ways 0-3 can be unlocked, and are overwritten by normal cache behavior.

Lockdown format C has a pattern-matching field, so any of the eight ways can be locked. There is no requirement to start at 0 and work up. If all ways are marked as being locked, then nothing is allocated.

Note

It is recommended that the cache lockdown register is modified with a read-modify-write sequence.

Replacement strategy

The cache controller uses a pseudo-random replacement strategy. A deterministic replacement strategy can be achieved, when you use them in combination with the lockdown registers.

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

If 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 so required, you can separate data and instructions into separate ways of the L2 cache.

3.3.16 Register 15, Test and Debug

The test registers enable the use of the cache data line and cache tag field to be read or written to directly by use of the secure transactions to the peripheral port. The debug register changes the behavior of the cache for debug purposes.

Note

If a test or a debug register is written to while a cache maintenance background task is not completed, it is ignored and returns a SLVERR response.

This section consists of:

- *Test Registers*
- *L2 Debug Control Register* on page 3-34.

Test Registers

The Test Operation Register enables the contents of the L2 cache to be read and written, depending on the **ARPROT[1]** or **AWPROT[1]** non-secure access tag. When test operations are in progress the peripheral port is not stalled, however if more accesses to the Test, Debug or Cache Maintenance Register are received, then they are ignored and a response of SLVERR is returned. Because the test operations are performed in the background, you must poll bit[0] of the test operation register to check for operation completion.

Secure test read

A secure test read operation is permitted to read both secure and NS cache lines. Write to the Test Operation Register with the nRW bit cleared. This enables the content of the line designated by the Way/Index fields is put in the Line Data Registers and its attributes put in the Line Tag Register. Tag each of the eight data line registers with the NS tag read from the tag line of the cache line read. All information required about the cache line can then be retrieved by reading Line Data and Tag registers, using secure accesses.

Secure test write:

A secure test write is permitted to write both secure and NS cache lines. Write all line data and attributes first in both the Line Data Registers and the Line Tag Register, using secure accesses. Tag as secure, each of the eight Line Data Registers. At this time, by writing to the Test Operation

Register, the cache line designated by the Way/Index fields can be updated with register contents. In particular, data is tagged in the cache with the NS bit of the line tag register.

Figure 3-17 shows the Test Operation Register bit assignments. The Test Operation Register is stored in Index/Way format:

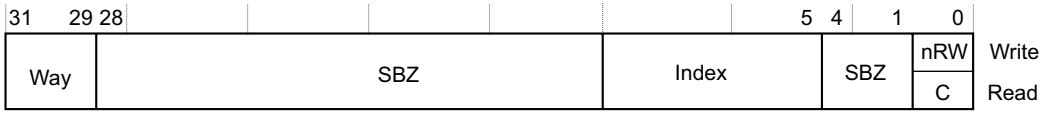


Figure 3-17 Test Operation Register

Note

The index field size depends on the cache way size.

Table 3-21 shows the register bit assignments.

Table 3-21 Test Operation Register

Bit	Field	Description
[31:29]	Way	-
[28:?]	SBZ	Depends on the index field size
[?:5]	Index	The index field size depends on the cache way size
[4:1]	SBZ	-
[1:0]	nRW/C	Write nRW, read bit C

Figure 3-18 shows the cache data line organization. In each cache line the words are stored in the figure:

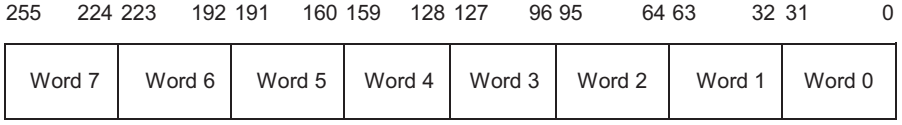


Figure 3-18 Cache data line organization

Figure 3-19 on page 3-33 shows the Line Tag Register mapping.

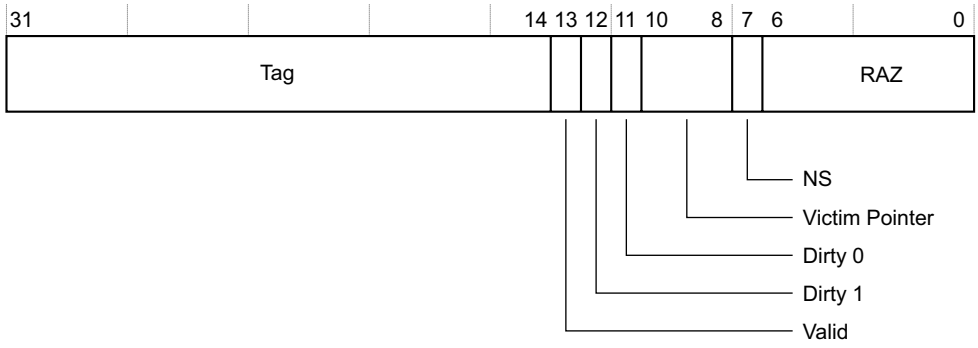


Figure 3-19 Line Tag Register format

Table 3-22 shows the register bit assignments.

Table 3-22 Line Tag Register format

Bit	Field	Description
[31:14]	Tag	-
[13]	Valid	-
[12]	Dirty 1	Defines the state of the last four words in the cache line, words 4 to 8.
[11]	Dirty 0	Defines the state of the first four words in the cache line, words 0 to 3.
[10:8]	Victim Pointer	Defines the last allocated way.
[7]	NS	-
[6:0]	RAZ	-

Note

- The Dirty 0 bit defines state of the first 4 words in the cache line, that is word 0 to 3.
- The Dirty 1 bit defines state of the last 4 words in the cache line, that is word 4 to 8.
- The Victim pointer field defines last allocated way.
- Address `BASE + 0xF10` stands for Word0 in the line, up to `Base + 0xF2C` that stands for Word7.

- If the RTL RAM models you are using do not initialize all RAM locations, there is a danger that simulation fails if the programmer attempts to read cache lines that have not been written to previously.

L2 Debug Control Register

The L2 Debug Control Register is used to force specific cache behavior required for debug. This register has read-only or read/write permission, and can be accessed by a secure access. If a NS access tries to read or write to this register, a DECERR error is returned on the write response channel and the register is not updated.

Figure 3-20 shows the L2 Debug Control Register.

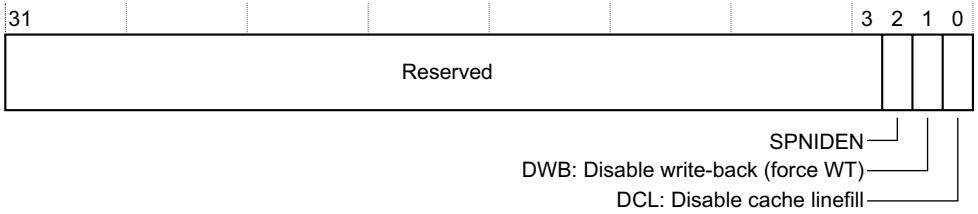


Figure 3-20 L2 Debug Control Register

Table 3-23 shows the L2 Debug Control Register bit assignments.

Table 3-23 Debug Control Register bit assignments

Bit	Field	Description
[31:3]	Reserved	SBZ / RAZ
[2]	SPNIDEN	Pollable read bit for **SPNIDEN**
[1]	DWB: Disable write-back (force WT)	0 = Enable write-back behavior (default) 1 = Force write-through behavior
[0]	DCL: Disable cache linefill	0 = Enable cache linefills (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 accesses 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. However, to achieve coherency, you can write back the other entries to the main memory.

Disabling cache linefills

If you set the DCL bit to 1, it prevents the cache from updating when performing a linefill on a miss. When set, a linefill is performed on a cache miss, reading eight words from external memory, but the cache is not updated with the linefill data. This mode of operation is required for debug so that the memory image, as seen by the 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.

Appendix A
Signal Descriptions

This appendix describes the L220 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	Source/destination voltage pool	Scaling mechanism
CLK	Input	Main clock and CPU side clock	V_{soc}	Level shifters
nRESET	Input	Global reset, active LOW	V_{soc}	Level shifters
CLKOUT	Output	Global clock signals for RAM	V_{ram}/V_{core}[a]	Level shifters

a. Dependent on the scaling ability of partner RAMs, if characterization data does not exist for RAM, they require level shifting to an independent voltage pool (that is, V_{ram}).

A.2 Configuration

Table A-2 shows the configuration signals.

Table A-2 Configuration signals

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
MASTNUM	Input	0 = 1 master port (**M1**) 1 = 2 master ports (**M0** and **M1**)	V_{core}	N/A
SLAVENUM	Input	0 = 1 slave port (**S1**) 1 = 2 slave ports (**S0** and **S1**)	V_{core}	N/A
CACHEID[5:0]	Input	L220 cache ID	V_{core}	N/A
SCANENABLE	Input	DFT Test enable, held during serial shift of scan chains and LOW for capture	V_{soc}/V_{core}[a]	Level shifters
DIRTYLAT[2:0]	Output	3'b111 default Latency for Dirty RAM from Auxiliary Control Register	V_{soc}	Level shifters
TAGLAT[2:0]	Output	3'b111 default Latency for tag RAM from Auxiliary Control Register	V_{soc}	Level shifters
WDATALAT[2:0]	Output	3'b111 default Latency for Write data RAM from Auxiliary Control Register	V_{soc}	Level shifters
RDATALAT[2:0]	Output	3'b111 default Latency for Read data RAM from Auxiliary Control Register	V_{soc}	Level shifters
L220CLAMP	Input	Enables the clamp cells between VDD Core and VDD SoC	V_{soc}	N/A
RAMCLAMP	Input	Enables the clamp cells in the dormant mode	V_{ram}/V_{core}	Level shifters

a. Voltage domain depends on test methodology and is implementation specific.

A.3 Slave, peripheral and master ports

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

- *Slave port 0*
- *Slave port 1* on page A-6
- *Peripheral port* on page A-8
- *Master port 0* on page A-10
- *Master port 1* on page A-12.

A.3.1 Slave port 0

Table A-3 shows the slave port 0 signals.

Table A-3 Slave port 0

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
ACLKENS0	Input	Clock enable, to indicate a valid rising edge when interfacing to synchronous clock domain of an integer division.	V_{core}	Level shifters
AWIDS0[5:0]	Input	Address identification	V_{core}	N/A
AWADDRS0[31:0]	Input	Address bus	V_{core}	N/A
AWLENS0[3:0]	Input	Burst length	V_{core}	N/A
AWSIZES0[2:0]	Input	Burst size	V_{core}	N/A
AWBURSTS0[1:0]	Input	Burst type	V_{core}	N/A
AWLOCKS0[1:0]	Input	Lock type	V_{core}	N/A
AWCACHES0[3:0]	Input	Cache information	V_{core}	N/A
AWPROTS0[2:0]	Input	Protection unit information	V_{core}	N/A
AWSIDEBANDS0[4:0]	Input	Sideband information	V_{core}	N/A
AWVALIDS0	Input	Address valid	V_{core}	N/A
AWREADYS0	Output	Address accepted	V_{core}	N/A
WRITEBACKS0[1:0]	Input	Indicates transaction is an eviction and clean/dirty status	V_{core}	N/A

Table A-3 Slave port 0 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
WIDS0[5:0]	Input	Write ID	V_{core}	N/A
WLASTS0	Input	Write last transfer	V_{core}	N/A
WDATAS0[63:0]	Input	Write data bus	V_{core}	N/A
WSTRBS0[7:0]	Input	Write strobes	V_{core}	N/A
WVALIDS0	Input	Write data valid	V_{core}	N/A
WREADYS0	Output	Write data accepted	V_{core}	N/A
BIDS0[5:0]	Output	Write identification	V_{core}	N/A
BRESPS0[1:0]	Output	Write response	V_{core}	N/A
BVALIDS0	Output	Write response valid	V_{core}	N/A
BREADYS0	Input	Write response accepted	V_{core}	N/A
ARIDS0[5:0]	Input	Address identification	V_{core}	N/A
ARADDRS0[31:0]	Input	Address bus	V_{core}	N/A
ARLENS0[3:0]	Input	Burst length	V_{core}	N/A
ARSIZES0[2:0]	Input	Burst size	V_{core}	N/A
ARBURSTS0[1:0]	Input	Burst type	V_{core}	N/A
ARLOCKS0[1:0]	Input	Lock type	V_{core}	N/A
ARCACHES0[3:0]	Input	Cache information	V_{core}	N/A
ARPROTS0[2:0]	Input	Protection unit information	V_{core}	N/A
ARSIDEBANDS0[4:0]	Input	Sideband information	V_{core}	N/A
ARVALIDS0	Input	Address valid	V_{core}	N/A
ARREADYS0	Output	Address accepted	V_{core}	N/A
RIDS0[5:0]	Output	Read identification	V_{core}	N/A
RLASTS0	Output	Read last transfer	V_{core}	N/A

Table A-3 Slave port 0 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
RDATAS0[63:0]	Output	Read data bus	V_{core}	N/A
RRESPS0[1:0]	Output	Read response	V_{core}	N/A
RVALIDS0	Output	Read data valid	V_{core}	N/A
RREADYS0	Input	Read accepted	V_{core}	N/A

A.3.2 Slave port 1

Table A-4 shows the slave port 1 signals.

Table A-4 Slave port 1

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
ACLKENS1	Input	Clock enable, to indicate a valid rising edge when interfacing to synchronous clock domain of an integer division	V_{core}	Level shifters
AWIDS1[5:0]	Input	Address identification	V_{core}	N/A
AWADDRS1[31:0]	Input	Address bus	V_{core}	N/A
AWLENS1[3:0]	Input	Burst length	V_{core}	N/A
AWSIZES1[2:0]	Input	Burst size	V_{core}	N/A
AWBURSTS1[1:0]	Input	Burst type	V_{core}	N/A
AWLOCKS1[1:0]	Input	Lock type	V_{core}	N/A
AWCACHES1[3:0]	Input	Cache information	V_{core}	N/A
AWPROTS1[2:0]	Input	Protection unit information	V_{core}	N/A
AWSIDEBANDS1[4:0]	Input	Sideband information	V_{core}	N/A
AWVALIDS1	Input	Address valid	V_{core}	N/A
AWREADYS1	Output	Address accepted	V_{core}	N/A

Table A-4 Slave port 1 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
WRITEBACKS1[1:0]	Input	Indicates transaction is an eviction and clean/dirty status	V_{core}	N/A
WIDS1[5:0]	Input	Write ID	V_{core}	N/A
WLASTS1	Input	Write last transfer	V_{core}	N/A
WDATAS1[63:0]	Input	Write data bus	V_{core}	N/A
WSTRBS1[7:0]	Input	Write strobes	V_{core}	N/A
WVALIDS1	Input	Write data valid	V_{core}	N/A
WREADYS1	Output	Write data accepted	V_{core}	N/A
BIDS1[5:0]	Output	Write identification	V_{core}	N/A
BRESPS1[1:0]	Output	Write response	V_{core}	N/A
BVALIDS1	Output	Write response valid	V_{core}	N/A
BREADYS1	Input	Write response accepted	V_{core}	N/A
ARIDS1[5:0]	Input	Address identification	V_{core}	N/A
ARADDRS1[31:0]	Input	Address bus	V_{core}	N/A
ARLENS1[3:0]	Input	Burst length	V_{core}	N/A
ARSIZES1[2:0]	Input	Burst size	V_{core}	N/A
ARBURSTS1[1:0]	Input	Burst type	V_{core}	N/A
ARLOCKS1[1:0]	Input	Lock type	V_{core}	N/A
ARCACHES1[3:0]	Input	Cache information	V_{core}	N/A
ARPROTS1[2:0]	Input	Protection unit information	V_{core}	N/A
ARSIDEBANDS1[4:0]	Input	Sideband information	V_{core}	N/A
ARVALIDS1	Input	Address valid	V_{core}	N/A
ARREADYS1	Output	Address accepted	V_{core}	N/A
RIDS1[5:0]	Output	Read identification	V_{core}	N/A

Table A-4 Slave port 1 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
RLASTS1	Output	Read last transfer	V_{core}	N/A
RDATAS1[63:0]	Output	Read data bus	V_{core}	N/A
RRESPS1[1:0]	Output	Read response	V_{core}	N/A
RVALIDS1	Output	Read data valid	V_{core}	N/A
RREADYS1	Output	Read accepted	V_{core}	N/A

A.3.3 Peripheral port

Table A-5 shows the peripheral port signals.

Table A-5 Peripheral port

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
ACLKP[a]	Input	Clock for the V_{soc} part of the peripheral port register slice	V_{soc}	N/A
ACLKENP	Input	Clock enable, to indicate a valid rising edge when interfacing to synchronous clock domain of an integer division	V_{soc}	Level shifters
ARESETPn[a]	Input	Reset for the V_{soc} part of the peripheral port register slice	V_{soc}	N/A
SYNCMODEREQP[a]	Input	Synchronous mode request signal from IEC to synchronize peripheral port	V_{soc}	N/A
SYNCMODEACKP[a]	Output	Synchronous mode acknowledge signal to IEC in the peripheral port	V_{soc}	N/A
AWIDP[7:0]	Input	Address Identification	V_{soc}	Register slice
AWADDRP[11:0]	Input	Address bus. This corresponds to address[11:2]	V_{soc}	Register slice
AWLENP[3:0]	Input	Burst length	V_{soc}	Register slice
AWSIZEP[2:0]	Input	Size	V_{soc}	Register slice

Table A-5 Peripheral port (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
AWBURSTP[1:0]	Input	Burst type	V_{soc}	Register slice
AWPROTP[2:0]	Input	Protection unit information	V_{soc}	Register slice
AWVALIDP	Input	Address valid	V_{soc}	Register slice
AWREADYP	Output	Address accepted	V_{soc}	Register slice
WIDP[7:0]	Input	Write Identification	V_{soc}	Register slice
WLASTP	Input	Write last transfer	V_{soc}	Register slice
WDATAP[31:0]	Input	Write data bus	V_{soc}	Register slice
WSTRBP[3:0]	Input	Byte lane strobe	V_{soc}	Register slice
WVALIDP	Input	Write data valid	V_{soc}	Register slice
WREADYP	Output	Write data accepted	V_{soc}	Register slice
BIDP[7:0]	Output	Write response identification	V_{soc}	Register slice
BRESPP[1:0]	Output	Write response	V_{soc}	Register slice
BVALIDP	Output	Write response valid	V_{soc}	Register slice
BREADYP	Input	Write response accepted	V_{soc}	Register slice
ARIDP[7:0]	Input	Address Identification	V_{soc}	Register slice
ARADDRP[11:0]	Input	Address bus, corresponding to address [11:2]	V_{soc}	Register slice
ARLENP[3:0]	Input	Burst length	V_{soc}	Register slice
ARSIZEP[2:0]	Input	Size	V_{soc}	Register slice
ARBURSTP[1:0]	Input	Burst type	V_{soc}	Register slice
ARPROTP[2:0]	Input	Protection unit information	V_{soc}	Register slice
ARVALIDP	Input	Address valid	V_{soc}	Register slice
ARREADYP	Output	Address accepted	V_{soc}	Register slice
RIDP[7:0]	Output	Read identification	V_{soc}	Register slice

Table A-5 Peripheral port (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
RLASTP	Output	Read last transfer	V_{soc}	Register slice
RDATAP[31:0]	Output	Read data bus	V_{soc}	Register slice
RRESPP[1:0]	Output	Read response	V_{soc}	Register slice
RVALIDP	Output	Read data valid	V_{soc}	Register slice
RREADYP	Input	Read accepted	V_{soc}	Register slice
CSYSREQ	Input	Input from system clock controller to put cache controller in low power state (AXI low-power interface)	V_{soc}	Level shifters
CSYSACK	Output	Output from cache controller to acknowledge both low power state request and the exit from low power state (AXI low-power interface).	V_{soc}	Level shifters
CACTIVE	Output	L220 not IDLE, low power mode is not active (AXI low-power interface).	V_{soc}	Level shifters

a. Ports are configurable, and are not present if IEM slices are disabled.

A.3.4 Master port 0

Table A-6 shows the master port 0 signals.

Table A-6 Master port 0

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
ACLKM0[a]	Input	Clock for the V_{soc} part of the **M0** port register slice	V_{soc}	N/A
ACLKENM0	Input	Clock enable, to indicate a valid rising edge when interfacing to synchronous clock domain of an integer division	V_{soc}	Level shifters
ARESETM0n[a]	Input	Reset for the V_{soc} part of the **M0** port register slice	V_{soc}	N/A

Table A-6 Master port 0 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
SYNCMODEREQM0[a]	Input	Synchronous mode request signal from IEC to synchronize **M0** port	V_{soc}	N/A
SYNCMODEACKM0[a]	Output	Synchronous mode acknowledge signal to IEC (**M0** port)	V_{soc}	N/A
AWIDM0[7:0]	Output	Address identification	V_{soc}	Register slice
AWADDRM0[31:0]	Output	Address bus	V_{soc}	Register slice
AWLENM0[3:0]	Output	Burst length	V_{soc}	Register slice
AWSIZEM0[2:0]	Output	Burst size	V_{soc}	Register slice
AWBURSTM0[1:0]	Output	Burst type	V_{soc}	Register slice
AWLOCKM0[1:0]	Output	Lock type	V_{soc}	Register slice
AWCACHEM0[3:0]	Output	Cache information	V_{soc}	Register slice
AWPROTM0[2:0]	Output	Protection unit information	V_{soc}	Register slice
AWSIDEBANDM0[4:0]	Output	Sideband information	V_{soc}	Register slice
AWVALIDM0	Output	Address valid	V_{soc}	Register slice
AWREADYM0	Input	Address accepted	V_{soc}	Register slice
WIDM0[7:0]	Output	Write ID	V_{soc}	Register slice
WLASTM0	Output	Write last transfer	V_{soc}	Register slice
WDATAM0[63:0]	Output	Write data bus	V_{soc}	Register slice
WSTRBM0[7:0]	Output	Write strobes	V_{soc}	Register slice
WVALIDM0	Output	Write data valid	V_{soc}	Register slice
WREADYM0	Input	Write data accepted	V_{soc}	Register slice
BIDM0[7:0]	Input	Write identification	V_{soc}	Register slice
BRESPM0[1:0]	Input	Write response	V_{soc}	Register slice
BVALIDM0	Input	Write response valid	V_{soc}	Register slice

Table A-6 Master port 0 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
BREADYM0	Output	Write response accepted	V_{soc}	Register slice
ARIDM0[7:0]	Output	Address identification	V_{soc}	Register slice
ARADDRM0[31:0]	Output	Address bus	V_{soc}	Register slice
ARLENM0[3:0]	Output	Burst length	V_{soc}	Register slice
ARSIZEM0[2:0]	Output	Burst size	V_{soc}	Register slice
ARBURSTM0[1:0]	Output	Burst type	V_{soc}	Register slice
ARLOCKM0[1:0]	Output	Lock type	V_{soc}	Register slice
ARCACHEM0[3:0]	Output	Cache information	V_{soc}	Register slice
ARPROTM0[2:0]	Output	Protection unit information	V_{soc}	Register slice
ARSIDEBANDM0[4:0]	Output	Sideband information	V_{soc}	Register slice
ARVALIDM0	Output	Address valid	V_{soc}	Register slice
ARREADYM0	Input	Address accepted	V_{soc}	Register slice
RIDM0[7:0]	Input	Read identification	V_{soc}	Register slice
RLASTM0	Input	Read last transfer	V_{soc}	Register slice
RDATAM0[63:0]	Input	Read data bus	V_{soc}	Register slice
RRESPM0[1:0]	Input	Read response	V_{soc}	Register slice
RVALIDM0	Input	Read data valid	V_{soc}	Register slice
RREADYM0	Output	Read accepted	V_{soc}	Register slice

a. Ports are configurable, and are not present if IEM slices are disabled.

A.3.5 Master port 1

Table A-7 on page A-13 shows the master port 1 signals.

Table A-7 Master port 1

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
ACLKM1[a]	Input	Clock for the V_{soc} part of the **M1** port register slice	V_{soc}	N/A
ACLKENM1	Input	Clock enable, to indicate a valid rising edge when interfacing to synchronous clock domain of an integer division	V_{soc}	Register shifters
ARESETM1n[a]	Input	Reset for the V_{soc} part of the **M1** port register slice	V_{soc}	N/A
SYNCMODEREQM1[a]	Input	Synchronous mode request signal from IEC to synchronize **M1** port	V_{soc}	N/A
SYNCMODEACKM1[a]	Output	Synchronous mode acknowledge signal to IEC (**M1** port)	V_{soc}	N/A
AWIDM1[7:0]	Output	Address identification	V_{soc}	Register slice
AWADDRM1[31:0]	Output	Address valid	V_{soc}	Register slice
AWLENM1[3:0]	Output	Burst length	V_{soc}	Register slice
AWSIZEM1[2:0]	Output	Burst size	V_{soc}	Register slice
AWBURSTM1[1:0]	Output	Burst type	V_{soc}	Register slice
AWLOCKM1[1:0]	Output	Lock type	V_{soc}	Register slice
AWCACHEM1[3:0]	Output	Cache information	V_{soc}	Register slice
AWPROTM1[2:0]	Output	Protection unit information	V_{soc}	Register slice
AWSIDEBANDM1[4:0]	Output	Sideband information	V_{soc}	Register slice
AWVALIDM1	Output	Address valid	V_{soc}	Register slice
AWREADYM1	Input	Address accepted	V_{soc}	Register slice
WIDM1[7:0]	Output	Write ID	V_{soc}	Register slice
WLASTM1	Output	Write last transfer	V_{soc}	Register slice
WDATAM1[63:0]	Output	Write data bus	V_{soc}	Register slice
WSTRBM1[7:0]	Output	Write strobes	V_{soc}	Register slice

Table A-7 Master port 1 (continued)

Signal	Type	Description	Source/ destination voltage pool	Scaling mechanism
WVALIDM1	Output	Write data valid	V_{soc}	Register slice
WREADYM1	Input	Write data accepted	V_{soc}	Register slice
BIDM1[7:0]	Input	Write identification	V_{soc}	Register slice
BRESPM1[1:0]	Input	Write response	V_{soc}	Register slice
BVALIDM1	Input	Write response valid	V_{soc}	Register slice
BREADYM1	Output	Write response accepted	V_{soc}	Register slice
ARIDM1[7:0]	Output	Address identification	V_{soc}	Register slice
ARADDRM1[31:0]	Output	Address bus	V_{soc}	Register slice
ARLENM1[3:0]	Output	Burst length	V_{soc}	Register slice
ARSIZEM1[2:0]	Output	Burst size	V_{soc}	Register slice
ARBURSTM1[1:0]	Output	Burst type	V_{soc}	Register slice
ARLOCKM1[1:0]	Output	Lock type	V_{soc}	Register slice
ARCACHEM1[3:0]	Output	Cache information	V_{soc}	Register slice
ARPROTM1[2:0]	Output	Protection unit information	V_{soc}	Register slice
ARSIDEBANDM1[4:0]	Output	Sideband information	V_{soc}	Register slice
ARVALIDM1	Output	Address valid	V_{soc}	Register slice
ARREADYM1	Input	Address accepted	V_{soc}	Register slice
RIDM1[7:0]	Input	Read identification	V_{soc}	Register slice
RLASTM1	Input	Read last transfer	V_{soc}	Register slice
RDATAM1[63:0]	Input	Read data bus	V_{soc}	Register slice
RRESPM1[1:0]	Input	Read response	V_{soc}	Register slice
RVALIDM1	Input	Read data valid	V_{soc}	Register slice
RREADYM1	Output	Read accepted	V_{soc}	Register slice

a. Ports are configurable, and are not present if IEM slices are disabled.

A.4 RAM interface

The RAM interface consists of three sub interfaces:

- *Data RAM interface*
- *Tag RAM interface* on page A-16
- *Dirty RAM interface signals* on page A-16.

A.4.1 Data RAM interface

Table A-8 shows the data RAM interface signals.

Table A-8 Data RAM interface signals

Signal	Type	Description	Source/destination voltage pool	Scaling mechanism
DATARD[255:0]	Input	Data RAM read data	V_{ram}/V_{core}	Level shifters
DATAPRD[31:0]	Input	Data RAM parity read data	V_{ram}/V_{core}	Level shifters
DATAERR	Input	Data RAM error	V_{ram}/V_{core}	Level shifters
DATAADDR[15:0]	Output	Data RAM address	V_{ram}/V_{core}	Level shifters
DATACS	Output	Data RAM chip select	V_{ram}/V_{core}	Level shifters
DATAEN[31:0]	Output	Data RAM byte write enables	V_{ram}/V_{core}	Level shifters
DATAnRW	Output	Data RAM write control signal	V_{ram}/V_{core}	Level shifters
DATAPEN[31:0]	Output	Data parity RAM byte write enables	V_{ram}/V_{core}	Level shifters
DATAPnRW	Output	Data parity RAM write control signal	V_{ram}/V_{core}	Level shifters
DATAWD[255:0]	Output	Data RAM write data	V_{ram}/V_{core}	Level shifters
DATAPWD[31:0]	Output	Data RAM parity write data	V_{ram}/V_{core}	Level shifters

A.4.2 Tag RAM interface

Table A-9 shows the tag RAM interface signals.

Table A-9 Tag RAM interface

Signal	Type	Description	Source/destination voltage pool	Scaling mechanism
TAGRD0[19:0]	Input	Tag RAM 0 read data	V_{ram}/V_{core}	Level shifters
TAGRD1[19:0]	Input	Tag RAM 1 read data	V_{ram}/V_{core}	Level shifters
TAGRD2[19:0]	Input	Tag RAM 2 read data	V_{ram}/V_{core}	Level shifters
TAGRD3[19:0]	Input	Tag RAM 3 read data	V_{ram}/V_{core}	Level shifters
TAGRD4[19:0]	Input	Tag RAM 4 read data	V_{ram}/V_{core}	Level shifters
TAGRD5[19:0]	Input	Tag RAM 5 read data	V_{ram}/V_{core}	Level shifters
TAGRD6[19:0]	Input	Tag RAM 6 read data	V_{ram}/V_{core}	Level shifters
TAGRD7[19:0]	Input	Tag RAM 7 read data	V_{ram}/V_{core}	Level shifters
TAGPRD[7:0]	Input	Tag RAM parity read data	V_{ram}/V_{core}	Level shifters
TAGERR[7:0]	Input	Tag RAM error	V_{ram}/V_{core}	Level shifters
TAGADDR[12:0]	Output	Tag RAM address	V_{ram}/V_{core}	Level shifters
TAGCS[7:0]	Output	Tag RAM chip selects	V_{ram}/V_{core}	Level shifters
TAGEN	Output	Tag RAM write enable	V_{ram}/V_{core}	Level shifters
TAGWD[19:0]	Output	Tag RAM write data	V_{ram}/V_{core}	Level shifters
TAGPWD	Output	Tag RAM parity write data	V_{ram}/V_{core}	Level shifters

A.4.3 Dirty RAM interface signals

Table A-10 on page A-17 shows the Dirty RAM interface signals.

Table A-10 Dirty RAM interface signals

Signal	Type	Description	Source/destination voltage pool	Scaling mechanism
DIRTYRD[15:0]	Input	Dirty RAM read data	V_{ram}/V_{core}	Level shifters
DIRTYCS	Output	Dirty RAM chip select	V_{ram}/V_{core}	Level shifters
DIRTYEN[15:0]	Output	Dirty RAM write enables	V_{ram}/V_{core}	Level shifters
DIRTYnRW	Output	Dirty RAM write control signal	V_{ram}/V_{core}	Level shifters
DIRTYWD[15:0]	Output	Dirty RAM write data	V_{ram}/V_{core}	Level shifters

A.5 Cache event monitoring

Table A-11 shows the cache event monitoring signals.

Table A-11 Cache event monitoring signals

Signal	Type	Description	Destination voltage pool	Scaling mechanism
CO	Output	Eviction (CastOUT) of a line or a half line from the L2 cache.	V_{core}[a]	N/A
DRHIT	Output	Data read hit in the L2 cache.	V_{core}[a]	N/A
DRREQ	Output	Data read lookup to the L2 cache. Results in a hit or miss.	V_{core}[a]	N/A
DWHIT	Output	Data write hit in the L2 cache.	V_{core}[a]	N/A
DWREQ	Output	Data write lookup to the L2 cache. Results in a hit or miss.	V_{core}[a]	N/A
DWTREQ	Output	Data write lookup to the L2 cache with Write-Through attribute. Results in a hit or miss.	V_{core}[a]	N/A
IRHIT	Output	Instruction read hit in the L2 cache.	V_{core}[a]	N/A
IRREQ	Output	Instruction read lookup to the L2 cache. Results in a hit or miss.	V_{core}[a]	N/A
SPNIDEN	Input	Secure privileged non-invasive debug enable.	V_{soc}	Level shifters
WA	Output	Allocation into the L2 cache caused by a write (with Write-Allocate attribute) miss.	V_{core}[a]	N/A

a. Because event signals are only valid for a single cycle, they can only be monitored within the scope of the V_{core} domain.

A.6 Cache interrupt

Table A-12 shows the cache interrupt signals.

Table A-12 Cache Interrupt signals

Signal	Type	Description	Destination voltage pool	Scaling mechanism
DECERRINTR	Output	Decode error received on master port from L3	V_{soc}	Level shifters
SLVERRINTR	Output	Slave error received on master port from L3	V_{soc}	Level shifters
ERRRDINTR	Output	Error on L2 data RAM read	V_{soc}	Level shifters
ERRRTINTR	Output	Error on L2 tag RAM read	V_{soc}	Level shifters
ERRWDINTR	Output	Error on L2 data RAM write	V_{soc}	Level shifters
ERRWTINTR	Output	Error on L2 data RAM write	V_{soc}	Level shifters
PARRDINTR	Output	Parity error on L2 data RAM read	V_{soc}	Level shifters
PARRTINTR	Output	Parity error on L2 tag RAM read	V_{soc}	Level shifters
ECNTRINTR	Output	Event Counter Overflow/Increment	V_{soc}	Level shifters
L2CCINTR	Output	Combined Interrupt Output	V_{soc}	Level shifters

A.7 MBIST interface

Table A-13 shows the MBIST interface signals.

Table A-13 MBIST interface signals

Signal	Type	Description	Destination voltage pool	Scaling mechanism
MBISTADDR[17:0]	Input	**MBIST address** **MBISTADDR[17:0]** is used for data RAM, two LSBs are used as a doubleword select **MBISTADDR[14:2]** is used for tag, and dirty RAMs	V_{core}	N/A
MBISTCE[10:0]	Input	**MBIST RAM chip select** **MBISTCE[0]** = data RAM chip select **MBISTCE[1]** = tag RAM 0 chip select **MBISTCE[2]** = tag RAM 1 chip select **MBISTCE[3]** = tag RAM 2 chip select **MBISTCE[4]** = tag RAM 3 chip select **MBISTCE[5]** = tag RAM 4 chip select **MBISTCE[6]** = tag RAM 5 chip select **MBISTCE[7]** = tag RAM 6 chip select **MBISTCE[8]** = tag RAM 7 chip select **MBISTCE[9]** = dirty RAM chip select **MBISTCE[10]** = data parity RAM chip select	V_{core}	N/A
MBISTDIN[63:0]	Input	**MBIST Data In** **MBISTDIN[63:0]** = MBIST data in for data RAM**MBISTDIN[20:0]** = MBIST data in for tag RAM **MBISTDIN[15:0]** = MBIST data in for dirty RAM	V_{core}	N/A
MBISTWE	Input	MBIST Write Enable	V_{core}	N/A
MTESTON	Input	MBIST Mode Enable	V_{core}	N/A
MBISTDCTL[12:0]	Input	MBIST Data out Multiplexor control	V_{core}	N/A
MBISTDOUT[63:0]	Output	**MBIST Data Out** **MBISTDOUT[63:0]** = MBIST data out for data RAM**MBISTDOUT[20:0]** = MBIST data out for tag RAM**MBISTDOUT[15:0]** = MBIST data out for dirty RAM	V_{core}	N/A

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 L220 Cache Controller signal applied to the SoC is provided as a percentage in the tables in this chapter. This appendix contains the following sections:

B.1 Reset and configuration signal timing parameters

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

Table B-1 Reset and configuration

Port name	I/O type	Maximum constraint
nRESETn	Input	20%
MASTNUM	Input	20%
SLAVENUM	Input	20%
CACHEID[5:0]	Input	20%
DIRTYLAT[2:0]	Output	50%
TAGLAT[2:0]	Output	50%
WDATALAT[2:0]	Output	50%
RDATALAT[2:0]	Output	50%
SCANENABLE	Input	20%
L220CLAMP	Input	20%
RAMCLAMP	Input	20%

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

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

Table B-2 Slave port 0 I/O

Port name	I/O type	Maximum constraint
ACLKBENS0	Input	40%
AWIDS0[5:0]	Input	50%
AWADDRS0[31:0]	Input	50%
AWLENS0[3:0]	Input	50%
AWSIZES0[2:0]	Input	50%
AWBURSTS0[1:0]	Input	50%
AWLOCKS0[1:0]	Input	50%
AWCACHES0[3:0]	Input	50%
AWPROTS0[2:0]	Input	50%
AWSIDEBANDS0[4:0]	Input	50%
AWVALIDS0	Input	50%
AWREADYS0	Output	70%
WRITEBACKS0[1:0]	Input	50%
WIDS0[5:0]	Input	50%
WLASTS0	Input	50%
WDATAS0[63:0]	Input	50%
WSTRBS0[7:0]	Input	50%
WVALIDS0	Input	50%
WREADYS0	Output	70%
BIDS0[5:0]	Output	70%
BRESPS0[1:0]	Output	70%
BVALIDS0	Output	70%
BREADYS0	Input	50%

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

Port name	I/O type	Maximum constraint
ARIDS0[5:0]	Input	50%
ARADDRS0[31:0]	Input	50%
ARLENS0[3:0]	Input	50%
ARSIZES0[2:0]	Input	50%
ARBURSTS0[1:0]	Input	50%
ARLOCKS0[1:0]	Input	50%
ARCACHES0[3:0]	Input	50%
ARPROTS0[2:0]	Input	50%
ARSIDEBANDS0[4:0]	Input	50%
ARVALIDS0	Input	50%
ARREADYS0	Output	70%
RIDS0[5:0]	Output	70%
RLASTS0	Output	70%
RDATAS0[63:0]	Output	70%
RRESPS0[1:0]	Output	60%
RVALIDS0	Output	70%
RREADYS0	Input	50%

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

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

Table B-3 Slave port 1 I/O

Port name	I/O type	Maximum constraint
ACLKBENS1	Input	40%
AWIDS1[5:0]	Input	50%
AWADDRS1[31:0]	Input	50%
AWLENS1[3:0]	Input	50%
AWSIZES1[2:0]	Input	50%
AWBURSTS1[1:0]	Input	50%
AWLOCKS1[1:0]	Input	50%
AWCACHES1[3:0]	Input	50%
AWPROTS1[2:0]	Input	50%
AWSIDEBANDS1[4:0]	Input	50%
AWVALIDS1	Input	50%
AWREADYS1	Output	70%
WRITEBACKS1[1:0]	Input	50%
WIDS1[5:0]	Input	50%
WLASTS1	Input	50%
WDATAS1[63:0]	Input	50%
WSTRBS1[7:0]	Input	50%
WVALIDS1	Input	50%
WREADYS1	Output	70%
BIDS1[5:0]	Output	70%
BRESPS1[1:0]	Output	70%
BVALIDS1	Output	70%
BREADYS1	Input	50%

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

Port name	I/O type	Maximum constraint
ARIDS1[5:0]	Input	50%
ARADDRS1[31:0]	Input	50%
ARLENS1[3:0]	Input	50%
ARSIZES1[2:0]	Input	50%
ARBURSTS1[1:0]	Input	50%
ARLOCKS1[1:0]	Input	50%
ARCACHES1[3:0]	Input	50%
ARPROTS1[2:0]	Input	50%
ARSIDEBANDS1[4:0]	Input	50%
ARVALIDS1	Input	50%
ARREADYS1	Output	70%
RIDS1[5:0]	Output	70%
RLASTS1	Output	70%
RDATAS1[63:0]	Output	70%
RRESPS1[1:0]	Output	60%
RVALIDS1	Output	70%
RREADYS1	Input	50%

B.4 Peripheral slave port signal timing parameters

Table B-4 shows the peripheral slave port signal timing parameters.

Table B-4 Peripheral slave port

Port name	I/O type	Maximum constraint
ACLKBENP	Input	40%
ARESETPn	Input	20%
SYNCMODEREQP	Input	60%
SYNCMODEACKP	Output	40%
AWIDP[7:0]	Input	70%
AWADDRP[11:0]	Input	70%
AWLENP[3:0]	Input	70%
AWSIZEP[2:0]	Input	70%
AWBURSTP[1:0]	Input	70%
AWPROTP[2:0]	Input	70%
AWVALIDP	Input	50%
AWREADYP	Output	70%
WIDP[7:0]	Input	70%
WLASTP	Input	70%
WDATAP[31:0]	Input	70%
WSTRBP[3:0]	Input	70%
WVALIDP	Input	50%
WREADYP	Output	70%
BIDP[7:0]	Output	70%
BRESPP[1:0]	Output	70%
BVALIDP	Output	70%
BREADYP	Input	50%
ARIDP[7:0]	Input	70%

Table B-4 Peripheral slave port (continued)

Port name	I/O type	Maximum constraint
ARADDRP[11:0]	Input	70%
ARLENP[3:0]	Input	70%
ARSIZEP[2:0]	Input	70%
ARBURSTP[1:0]	Input	70%
ARPROTP[2:0]	Input	70%
ARVALIDP	Input	50%
ARREADYP	Output	70%
RIDP[7:0]	Output	70%
RLASTP	Output	70%
RDATAP[31:0]	Output	70%
RRESPP[1:0]	Output	70%
RVALIDP	Output	70%
RREADYP	Input	50%
CACTIVE	Output	60%
CSYSREQ	Input	70%
CSYSACK	Output	60%

B.5 Master port 0 I/O signal timing parameters

Table B-5 shows the master port 0 I/O signal timing parameters.

Table B-5 Master port 0 I/O

Port name	I/O type	Maximum constraint
ACLKBENM0	Input	60%
ARESETM0n	Input	20%
SYNCMODEREQM0	Input	60%
SYNCMODEACKM0	Output	40%
AWIDM0[7:0]	Output	60%
AWADDRM0[31:0]	Output	60%
AWLENM0[3:0]	Output	60%
AWSIZEM0[2:0]	Output	60%
AWBURSTM0[1:0]	Output	60%
AWLOCKM0[1:0]	Output	60%
AWCACHEM0[3:0]	Output	60%
AWPROTM0[2:0]	Output	60%
AWSIDEBANDM0[4:0]	Output	60%
AWVALIDM0	Output	60%
AWREADYM0	Input	50%
WIDM0[7:0]	Output	60%
WLASTM0	Output	60%
WDATAM0[63:0]	Output	60%
WSTRBM0[7:0]	Output	60%
WVALIDM0	Output	60%
WREADYM0	Input	50%
BIDM0[7:0]	Input	50%
BRESPM0[1:0]	Input	50%

Table B-5 Master port 0 I/O (continued)

Port name	I/O type	Maximum constraint
BVALIDM0	Input	50%
BREADYM0	Output	60%
ARIDM0[7:0]	Output	60%
ARADDRM0[31:0]	Output	60%
ARLENM0[3:0]	Output	60%
ARSIZEM0[2:0]	Output	60%
ARBURSTM0[1:0]	Output	60%
ARLOCKM0[1:0]	Output	60%
ARCACHEM0[3:0]	Output	60%
ARPROTM0[2:0]	Output	60%
ARSIDEBANDM0[4:0]	Output	60%
ARVALIDM0	Output	60%
ARREADYM0	Input	50%
RIDM0[7:0]	Input	50%
RLASTM0	Input	50%
RDATAM0[63:0]	Input	50%
RRESPM0[1:0]	Input	50%
RVALIDM0	Input	50%
RREADTYM0	Output	60%

B.6 Master port 1 I/O signal timing parameters

Table B-6 shows the master port 1 I/O signal timing parameters.

Table B-6 Master port 1 I/O

Port name	I/O type	Maximum constraint
ACLKBENM1	Input	60%
ARESETM1n	Input	20%
SYNCMODEREQM1	Input	60%
SYNCMODEACKM1	Output	40%
AWIDM1[7:0]	Output	60%
AWADDRM1[31:0]	Output	60%
AWLENM1[3:0]	Output	60%
AWSIZEM1[2:0]	Output	60%
AWBURSTM1[1:0]	Output	60%
AWLOCKM1[1:0]	Output	60%
AWCACHEM1[3:0]	Output	60%
AWPROTM1[2:0]	Output	60%
AWSIDEBANDM1[4:0]	Output	60%
AWVALIDM1	Output	60%
AWREADYM1	Input	50%
WIDM1[7:0]	Output	60%
WLASTM1	Output	60%
WDATAM1[63:0]	Output	60%
WSTRBM1[7:0]	Output	60%
WVALIDM1	Output	60%
WREADYM1	Input	50%
BIDM1[7:0]	Input	50%
BRESPM1[1:0]	Input	50%

Table B-6 Master port 1 I/O (continued)

Port name	I/O type	Maximum constraint
BVALIDM1	Input	50%
BREADYM1	Output	60%
ARIDM1[7:0]	Output	60%
ARADDRM1[31:0]	Output	60%
ARLENM1[3:0]	Output	60%
ARSIZEM1[2:0]	Output	60%
ARBURSTM1[1:0]	Output	60%
ARLOCKM1[1:0]	Output	60%
ARCACHEM1[3:0]	Output	60%
ARPROTM1[2:0]	Output	60%
ARSIDEBANDM1[4:0]	Output	60%
ARVALIDM1	Output	60%
ARREADYM1	Input	50%
RIDM1[7:0]	Input	50%
RLASTM1	Input	50%
RDATAM1[63:0]	Input	50%
RRESPM1[1:0]	Input	50%
RVALIDM1	Input	50%
RREADYM1	Output	60%

B.7 RAMs signal timing parameters

This section shows the RAMs signal timing parameters in:

- *Data RAM*
- *Tag RAM* on page B-14
- *Dirty RAM* on page B-14.

B.7.1 Data RAM

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

Table B-7 Data RAM

Port name	I/O type	Maximum constraint
DATARD[255:0]	Input	55%
DATAPRD[31:0]	Input	60%
DATAERR	Input	45%
DATAADDR[15:0]	Output	60%
DATACS	Output	60%
DATAEN[31:0]	Output	60%
DATAPEN[31:0]	Output	60%
DATAWD[255:0]	Output	60%
DATAPWD[31:0]	Output	60%
DATAnRW	Output	60%
DATAPnRW	Output	60%

B.7.2 Tag RAM

Table B-8 shows the Tag RAM signal timing parameters.

Table B-8 Tag RAM

Port name	I/O type	Maximum constraint
TAGRD0[19:0]	Input	70%
TAGRD1[19:0]	Input	70%
TAGRD2[19:0]	Input	70%
TAGRD3[19:0]	Input	70%
TAGRD4[19:0]	Input	70%
TAGRD5[19:0]	Input	70%
TAGRD6[19:0]	Input	70%
TAGRD7[19:0]	Input	70%
TAGERR[7:0]	Input	70%
TAGADDR[12:0]	Output	60%
TAGCS[7:0]	Output	60%
TAGEN	Output	60%
TAGWD[19:0]	Output	60%
TAGPWD	Output	60%
TAGPRD[7:0]	Input	70%

B.7.3 Dirty RAM

Table B-9 shows the Tag RAM signal timing parameters.

Table B-9 Dirty RAM

Port name	I/O type	Maximum constraint
DIRTYRD[15:0]	Input	70%
DIRTYCS	Output	60%
DIRTYEN[15:0]	Output	60%
DIRTYnRW	Output	60%
DIRTYWD[15:0]	Output	60%

B.8 Event monitor signal timing parameters

Table B-10 shows the event monitor signal timing parameters.

Table B-10 Event monitor

Port name	I/O type	Maximum constraint
CO	Output	40%
DRHIT	Output	40%
DRREQ	Output	40%
DWHIT	Output	40%
DWREQ	Output	40%
DWTREQ	Output	40%
IRHIT	Output	40%
IRREQ	Output	40%
SPNIDEN	Input	40%
WA	Output	40%

B.9 Cache interrupt ports signal timing parameters

Table B-11 shows the cache interrupt ports signal timing parameters.

Table B-11 Cache interrupt ports

Port name	I/O type	Maximum constraint
DECERRINTR	Output	50%
SLVERRINTR	Output	50%
ERRRDINTR	Output	50%
ERRRTINTR	Output	50%
ERRWDINTR	Output	50%
ERRWTINTR	Output	50%
PARRDINTR	Output	50%
PARRTINTR	Output	50%
ECNTRINTR	Output	50%
L2CCINTR	Output	50%

B.10 MBIST interface signal timing parameters

Table B-12 shows the MBIST interface signal timing parameters.

Table B-12 MBIST interface signal

Port name	I/O type	Maximum constraint
MBISTADDR[17:0]	Input	30%
MBISTCE[10:0]	Input	30%
MBISTDIN[63:0]	Input	30%
MBISTWE	Input	30%
MTESTON	Input	30%
MBISTDCTL[12:0]	Input	30%
MBISTDOUT[63:0]	Output	70%

Appendix C
Timing Diagrams

This appendix describes the timings of typical ARM L220 Cache Controller operations. It contains the following sections:

C.1 Single read hit transaction

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

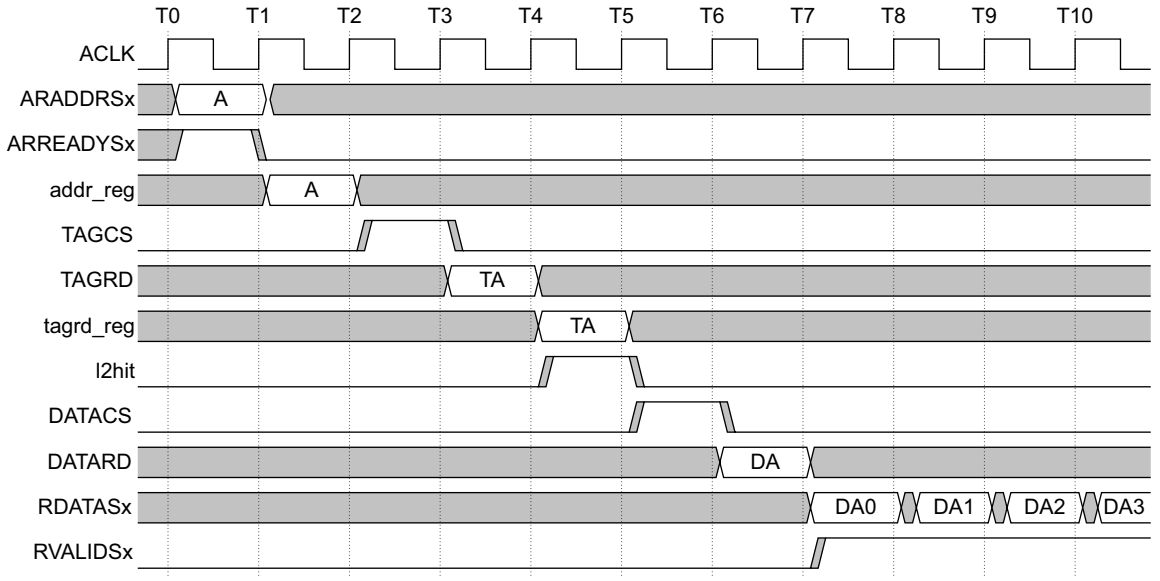


Figure C-1 Single read hit transaction

C.2 Single read miss transaction

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

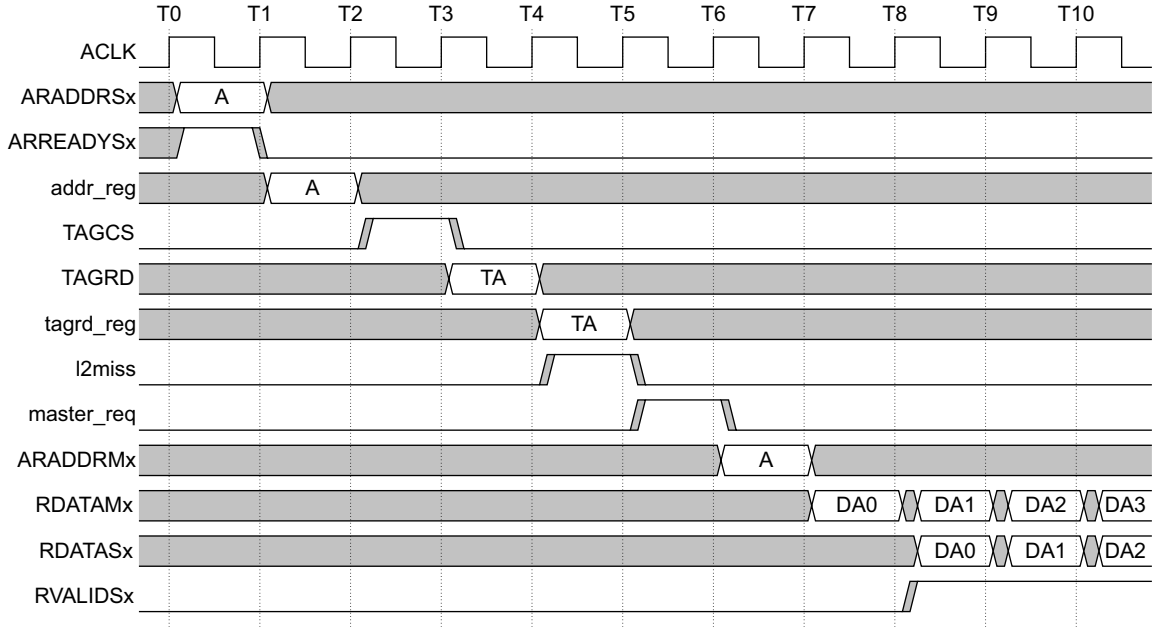


Figure C-2 Single read miss transaction

C.3 Two simultaneous read hits

Figure C-3 shows the timing for two simultaneous read hits.

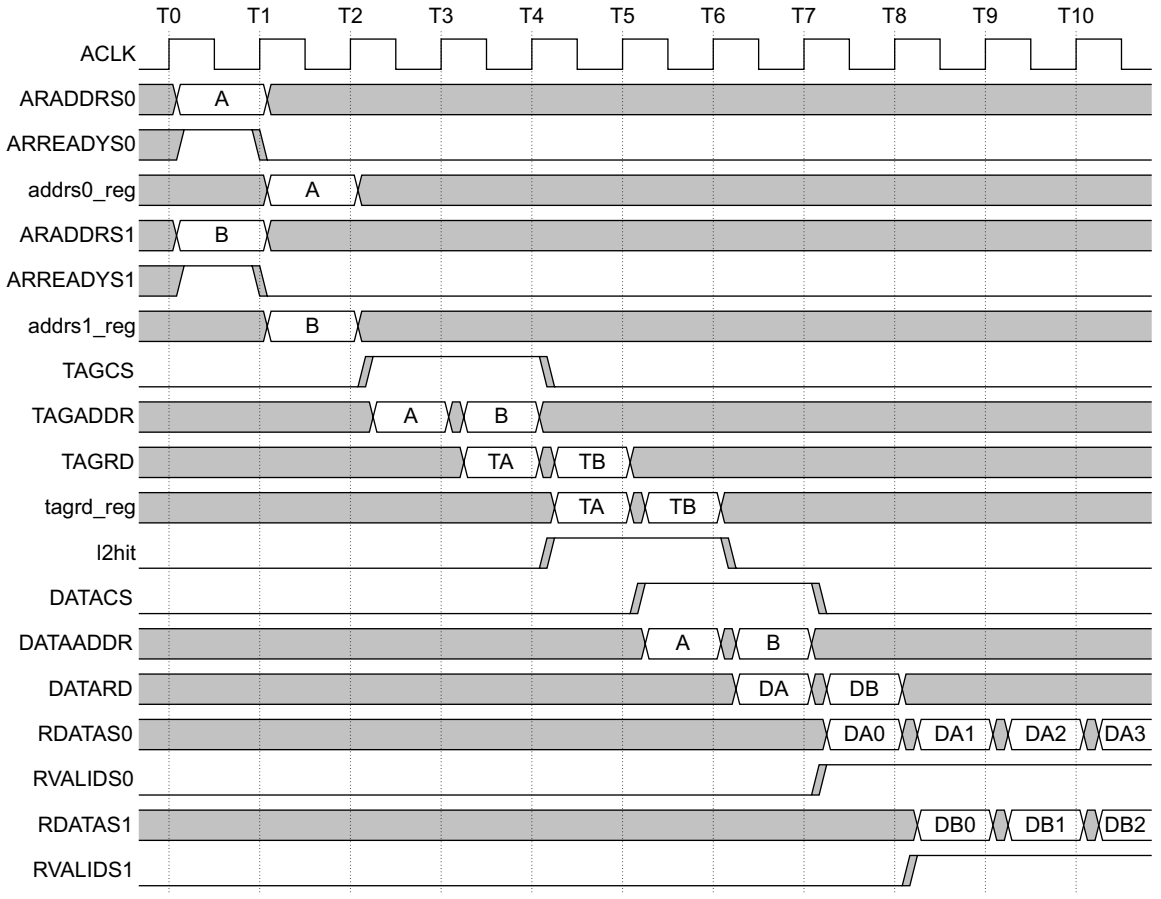


Figure C-3 Two simultaneous read hits

C.4 Single noncacheable read transaction

Figure C-4 shows the timing for a single noncacheable read transaction.

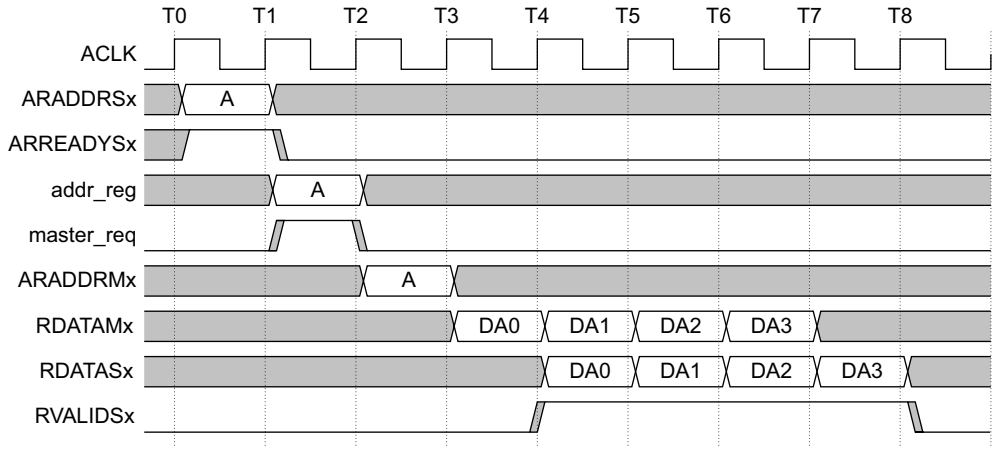


Figure C-4 Single noncacheable read transaction

C.5 Single bufferable write transaction

Figure C-5 shows the timing for a single bufferable write transaction.

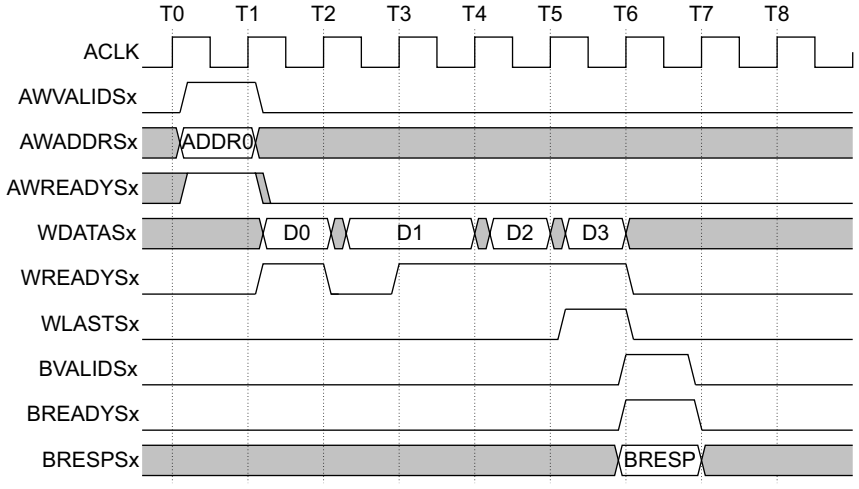


Figure C-5 Single bufferable write transaction

C.6 Single nonbufferable write transaction

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

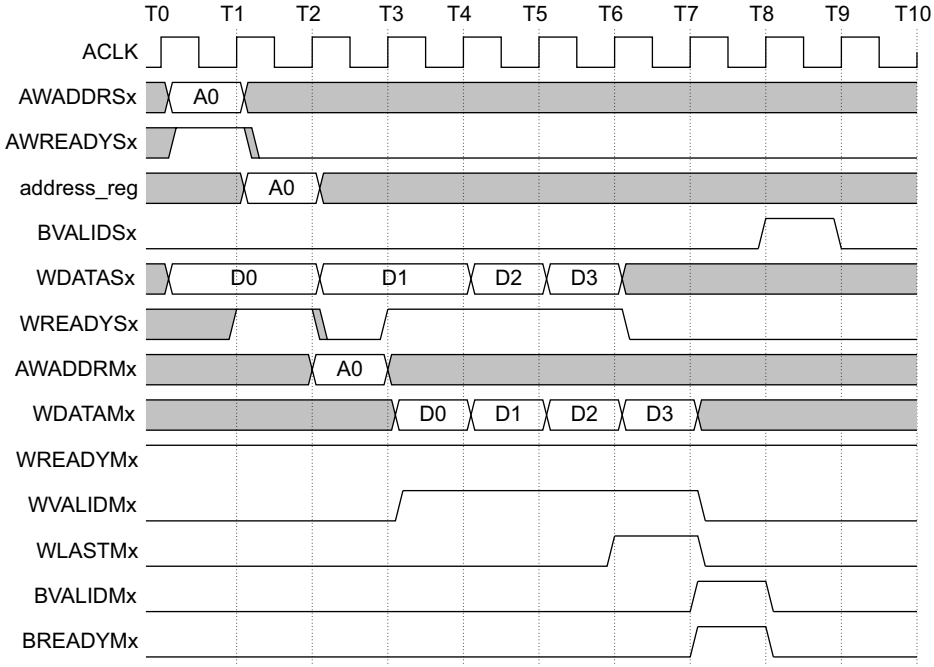


Figure C-6 Single nonbufferable write transaction

Glossary

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

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

See also Data Abort.

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

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

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

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

AMBA *See* Advanced Microcontroller Bus Architecture.

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

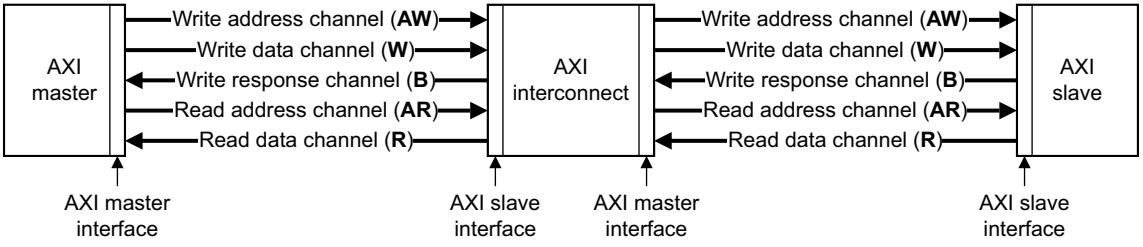
Architecture The organization of hardware and/or software that characterizes a processor and its attached components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture, ARMv6 architecture.

ASIC *See* Application Specific Integrated Circuit.

AXI *See* Advanced eXtensible Interface.

AXI channel order and interfaces
The block diagram shows:

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



AXI terminology

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

Active read transaction

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

Active transfer

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

Active write transaction

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

Completed transfer

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

Payload The non-handshake signals in a transfer.

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

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

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

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

Combined issuing capability

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

---

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

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

Read ID capability

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

Read ID width

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

Read issuing capability

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

Write ID capability

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

Write ID width

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

Write interleave capability

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

Write issuing capability

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

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

Combined acceptance capability

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

Read acceptance capability

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

Read data reordering depth

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

Write acceptance capability

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

Write interleave depth

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

Beat

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

See also Burst.

Big-endian

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

See also Little-endian and Endianness.

Big-endian memory

Memory in which:

- a byte or halfword at a word-aligned address is the most significant byte or halfword within the word at that address
- a byte at a halfword-aligned address is the most significant byte within the halfword at that address.

See also Little-endian memory.

Block address

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

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

Burst

A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

See also Beat.

Byte

An 8-bit data item.

Byte lane strobe

A signal that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of this signal corresponds to eight bits of the data bus.

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

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

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

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

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

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

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

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

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

Cache set
A cache set is a group of cache lines (or blocks). A set contains all the ways that can be addressed with the same index. The number of cache sets is always a power of two. All sets are accessed in parallel during a cache look-up.

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

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

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

Cast out
See Victim.

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

See also Dirty.

Coprocessor A processor that supplements the main processor. It carries out additional functions that the main processor cannot perform. Usually used for floating-point math calculations, signal processing, or memory management.

Copy back *See* Write-back.

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

See also Abort.

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

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

See also Clean.

Doubleword A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.

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

See also Little-endian and Big-endian

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

See also Direct-mapped cache.

Halfword A 16-bit data item.

IEM *See* Intelligent Energy Management.

Index *See* Cache index.

Index register A register specified in some load or store instructions. The value of this register is used as an offset to be added to or subtracted from the base register value to form the virtual address, which is sent to memory. Some addressing modes optionally enable the index register value to be shifted prior to the addition or subtraction.

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

Intelligent Energy Management (IEM)
A technology that enables dynamic voltage scaling and clock frequency variation to be used to reduce power consumption in a device.

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

Line *See* Cache line.

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

See also Big-endian and Endianness.

Little-endian memory
Memory in which:

- a byte or halfword at a word-aligned address is the least significant byte or halfword within the word at that address
- a byte at a halfword-aligned address is the least significant byte within the halfword at that address.

See also Big-endian memory.

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

Memory bank One of two or more parallel divisions of interleaved memory, usually one word wide, that enable reads and writes of multiple words at a time, rather than single words. All memory banks are addressed simultaneously and a bank enable or chip select signal determines which of the banks is accessed for each transfer. Accesses to sequential word addresses cause accesses to sequential banks. This enables the delays associated with accessing a bank to occur during the access to its adjacent bank, speeding up memory transfers.

Microprocessor *See* Processor.

Miss *See* Cache miss.

Processor A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Physical Address (PA)

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

See also Fast Context Switch Extension.

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

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

RAZ Read As Zero.

SBO *See* Should Be One.

SBZ *See* Should Be Zero.

SBZP *See* Should Be Zero or Preserved.

Set *See* Cache set.

Set-associative cache

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

Should Be One (SBO)

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

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

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

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

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

Tightly coupled memory (TCM)
: An area of low latency memory that provides predictable instruction execution or data load timing in cases where deterministic performance is required. TCMs are suited to holding: - critical routines (such as for interrupt handling) - scratchpad data - data types whose locality is not suited to caching - critical data structures (such as interrupt stacks).

TrustZone
: This is a security extension for the ARM architecture.

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

  See also Aligned.

UNP
: *See* Unpredictable.

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

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

Way
: *See* Cache way.

WB
: *See* Write-back.

Word
: A 32-bit data item.

Write
: Writes are defined as operations that have the semantics of a store. That is, the ARM instructions SRS, STM, STRD, STC, STRT, STRH, STRB, STRBT, STREX, SWP, and SWPB, and the Thumb instructions STM, STR, STRH, STRB, and PUSH. Java

instructions that are accelerated by hardware can cause a number of writes to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.

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

Write buffer A block of high-speed memory, arranged as a FIFO buffer, between the data cache and main memory, whose purpose is to optimize stores to main memory.

Write completion The memory system indicates to the processor that a write has been completed at a point in the transaction where the memory system is able to guarantee that the effect of the write is visible to all processors in the system. This is not the case if the write is associated with a memory synchronization primitive, or is to a Device or Strongly Ordered region. In these cases the memory system might only indicate completion of the write when the access has affected the state of the target, unless it is impossible to distinguish between having the effect of the write visible and having the state of target updated.

This stricter requirement for some types of memory ensures that any side-effects of the memory access can be guaranteed by the processor to have taken place. You can use this to prevent the starting of a subsequent operation in the program order until the side-effects are visible.

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

WT *See* Write-through.

Cache terminology diagram

The diagram below illustrates the following cache terminology:

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