# PL310 Cache Controller

**Revision: r0p0**

## Technical Reference Manual

ARM®


ARM DDI 0246A

# PL310 Cache Controller
## Technical Reference Manual



### Release Information

**Change history**

| Date | Issue | Confidentiality | Change |
|---|---|---|---|
| 30 November 2007 | A | Non-Confidential | First release for r0p0 |

### Proprietary Notice



### Confidentiality Status

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

### Product Status

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

### Web Address

`http://www.arm.com`

# Contents
# PL310 Cache Controller Technical Reference Manual

# List of Tables
# PL310 Cache Controller Technical Reference Manual

# List of Figures
# PL310 Cache Controller Technical Reference Manual

# Preface

This preface introduces the *PL310 Cache Controller Revision r0p0 Technical Reference Manual*. It contains the following sections:

## About this manual

This is the *Technical Reference Manual* (TRM) for the PL310 Cache Controller. In this manual the generic term cache controller means the PL310 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 PL310 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 the Glossary for definitions of terms used in this manual.

## Conventions

Conventions that this manual can use are described in:

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

### Typographical

The typographical conventions are:

***italic*** Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

**bold** Highlights interface elements, such as menu names. Denotes 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.

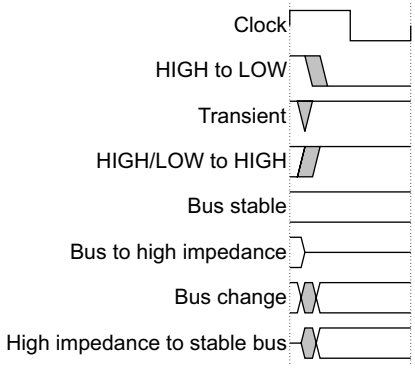
**< and >** Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example:

- `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* on page xiv explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

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




**Key to timing diagram conventions**

## Signals

The signal conventions are:

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

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

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

**Prefix AR** Denotes AXI read address channel signals.

**Prefix AW** Denotes AXI write address channel signals.

**Prefix B** Denotes AXI write response channel signals.

**Prefix C** Denotes AXI low-power interface signals.

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

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

**Prefix R** Denotes AXI read data channel signals.

**Prefix W** Denotes AXI write data channel signals.

### Numbering

The numbering convention is:

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

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

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

## Additional 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 cache controller. See the following documents for other relevant information:

- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *ARM Architecture Reference Manual* (ARM DDI 0406)
- *ARM PL310 MBIST Controller Technical Reference Manual* (ARM DDI 0402)
- *ARM PL310 Cache Controller Implementation Guide* (ARM DII 0045).

## Feedback

ARM welcomes feedback on the cache controller and its documentation.

### Feedback on this product

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

- the product name
- 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 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.

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

**Table 1-1 Typical memory sizes and access times**

| Memory type | Typical size | Typical access time |
|---|---|---|
| Processor registers | 128B | 1 cycle |
| On-chip L1 cache | 32KB | 1-2 cycle(s) |
| On-chip L2 cache | 256KB | 8 cycles |
| Main memory (L3) dynamic RAM | MB or GB[a] | 30-100 cycles |
| Back-up memory (hard disk) (L4) | MB or GB | > 500 cycles |

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

The cache controller features:

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

The cache controller is a unified, physically addressed, physically tagged cache with up to 16 ways. You can lock the replacement algorithm on a way basis, enabling the associativity to be reduced from 16-way down to one-way, direct mapped. The cache controller does not have snooping hardware to maintain coherency between caches, so you must maintain coherency by software.

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

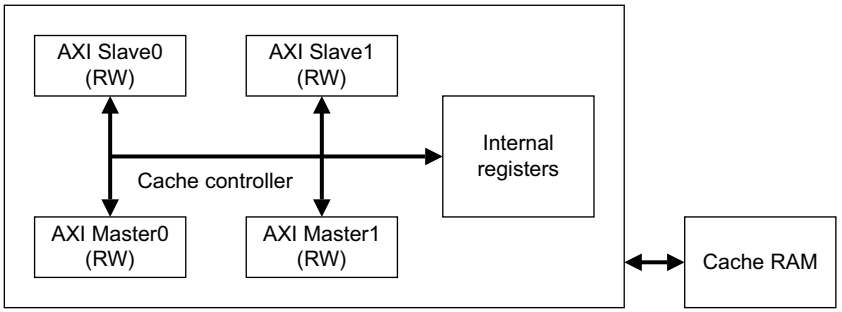



**Figure 1-1 Top level diagram**

### 1.1.1 Features

The cache controller has the following features:

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

  **Note**

  Lockdown format C is also known as way locking.

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

- Non-cacheable shared reads are treated as cacheable non-allocatable. Non-cacheable shared writes are treated as cacheable write-through no write-allocate. There is an option to override this behavior, 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 security level of access to L3 for eviction and WB.
  - Restrictions for NS accesses for control, configuration, and maintenance registers to restrict access to secure data.
- Critical word first linefill supported.
- Pseudo-Random victim selection policy. You can make this deterministic with use of lockdown registers.
- Four 256-bit *Line Fill Buffers* (LFBs), shared by the master ports. These buffers capture linefill data from main memory, waiting for a complete line before writing to L2 cache memory.
- Two 256-bit *Line Read Buffers* (LRBs) for each slave port. These buffers hold a line from the L2 memory in case of cache hit.
- Three 256-bit *Eviction Buffers* (EBs). These buffers hold evicted lines from the L2 cache, to be written back to main memory.
- Three 256-bit *Store Buffers* (STBs). These buffers hold bufferable writes before their draining to main memory, or the L2 cache. They enable multiple writes to the same line to be merged.
- Supports outstanding accesses on slave and master ports.
- Option to select one or two master ports.
- Option to select one or two slave ports.

  **Note**

  If only one slave is supported it is intended that only one master is configured.

- Software option to enable exclusive cache configuration, see *Cache operation* on page 2-5 for more information.
- Prefetching capability. See *Register 1, Auxiliary Control Register* on page 3-11 for more information.

- Support for integer (1:1, 2:1,...) and half-integer (1.5:1, 2.5:1,...) clock ratios controlled by clock enable inputs on slave and master ports.
- Wait, 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 also offered in the cache controller with two programmable 32-bit counters. Secure event and performance signals are only available when the signal on the **SPNIDEN** pin is configured HIGH.
- Configuration registers accessible using address decoding in the slave ports.
- Address filtering in the master ports enabling redirection of a certain address range to one master port while all other addresses are redirected to the other one.

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

**Table 1-2 RTL options**

| Feature | RTL option |
|---|---|
| 16-way associativity | `pl310_16_WAYS` |
| Number of slave ports | `pl310_S1` |
| Number of master ports | `pl310_M1` |
| Parity | `pl310_PARITY` |
| Lockdown by master | `pl310_LOCKDOWN_BY_MASTER` |
| Lockdown by line | `pl310_LOCKDOWN_BY_LINE` |
| Slave AXI ID width | `pl310_AXI_ID_MAX` |

**Note**

These options must be defined prior to synthesis.

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.

## 1.2 Typical system configuration

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

The cache controller supports:

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

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

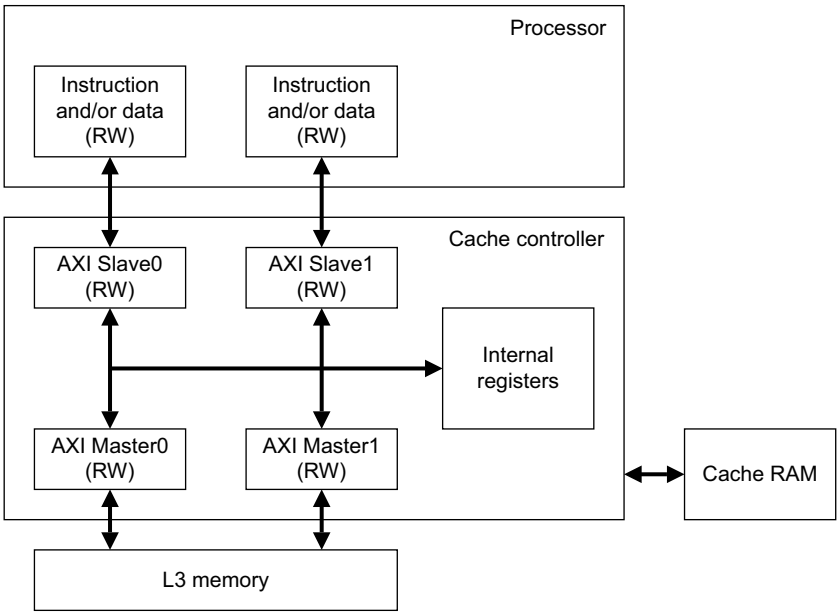



**Figure 1-2 Example cache controller interfaced to an ARM processor**

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

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

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

**Note**

- Table 1-3 does not take address filtering into account.
- In a one master port system, master port 1 is not implemented. All master port 0 transactions apply to both **S0** and **S1**.

## 1.3 Product revisions

This is the first release of this manual.

# Chapter 2
# **Functional Overview**

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

## 2.1 AXI master and slave interfaces

Table 2-1 shows the AXI master interface attributes.

**Table 2-1 AXI master interface attributes**

| Attribute | Value |
|---|---|
| Write issuing capability | 8, consisting of:<br>• 3 evictions<br>• 3 writes from store buffer<br>• 1 non-bufferable write from S0<br>• 1 non-bufferable write from S1. |
| Read issuing capability | 11, consisting of:<br>• 4 reads from S0<br>• 4 reads from S1<br>• 3 reads from store buffer. |
| Combined issuing capability | 25. |
| Write interleave capability | 1. |
| Write ID width | Parameterizable, default is 8. |
| Read ID width | Parameterizable, default is 8. |

The values in Table 2-1 are for the cache controller as a whole.

Table 2-2 shows the AXI slave interface attributes

**Table 2-2 AXI slave interface attributes**

| Attribute | Value per slave |
|---|---|
| Write acceptance capability | 1 |
| Read acceptance capability | 4 |
| Combined acceptance capability | 5 |
| Write interleave depth | 1 |
| Read data reorder depth | 4 |
| Write ID width | Parameterizable, defined by `pl310_AXI_ID_MAX` variable, default is 6 |
| Read ID width | Parameterizable, defined by `pl310_AXI_ID_MAX` variable, default is 6 |

### 2.1.1 Clock enable usage model in the cache controller AXI interfaces

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

Figure 2-1 shows how the different clock enable signals can be generated to support half integer clock ratios. In the figure, **ACLK** is the clock of the L3 AXI system.

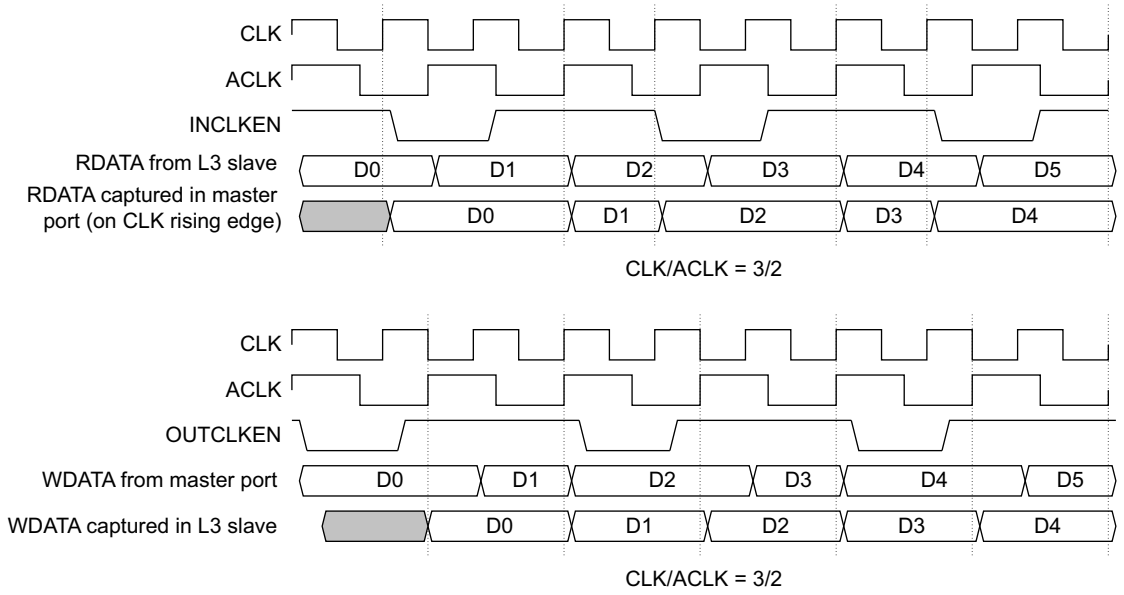



**Figure 2-1 Clock enable usage model for 1.5:1 clock ratio in master port**

See Table A-1 on page A-2 for a description of the clock enables. These clock enables 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 **INCLKEN** is HIGH.
- Outputs are updated on rising edges of **CLK** only when **OUTCLKEN** is HIGH.

## 2.2 Cache attributes

Table 2-3 describes the **AWCACHE[3:0]** and **ARCACHE[3:0]** signals as described in the *AMBA AXI Protocol Specification*, and the ARMv6 and ARMv7 equivalent meaning.

**Table 2-3 AWCACHE and ARCACHE definitions**

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

**Note**

- The shared attribute **AW/RUSERSx[0]** is not described in this table. Its behavior is explained in *Cache operation* on page 2-5, under *Shared attribute*.
- The cache controller supports all AXI cache attributes, even if the processor does not use all of them
- If the cache controller receives cacheable fixed transactions, **AWBURST/ARBURSTSx** = 00, the results are unpredictable.
- Table 2-3 does not show AXI locked and exclusive accesses.

## 2.3 Cache operation

Table 2-4 shows the general behavior of the cache controller depending on ARMv6 and ARMv7 transactions.

**Table 2-4 Cache operation descriptions**

| AXI transaction | ARMv6 and ARMv7 memory type attribute | Cache controller behavior |
|---|---|---|
| Noncacheable, nonbufferable | Strongly ordered | Read: Not cached in L2, results in L3 access.<br>Write: Not buffered, results in L3 access. |
| Bufferable only | Device | Read: Not cached in L2, results in L3 access.<br>Write: Not buffered, results in L3 access. |
| Cacheable but do not allocate | Outer non cacheable | Read: Not cached in L2, results in L3 access.<br>Write: Put in store buffer, write to L3 when store buffer 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.<br>Read miss: Linefill to L2.<br>Write hit: Put in store buffer, write to L2 and L3 when store buffer is drained.<br>Write miss: Put in store buffer, write to L3 when store buffer is drained. |
| Cacheable write-back, allocate on read | Outer write-back, no write allocate | Read hit: Read from L2.<br>Read miss: Linefill to L2.<br>Write hit: Put in store buffer, write to the L2 when store buffer is drained, mark line as dirty.<br>Write miss: Put in store buffer, write to L3 when store buffer is drained. |
| Cacheable write-through, allocate on write | | Read hit: Read from L2.<br>Read miss: Not cached in L2, causes L3 access.<br>Write hit: Put in store buffer, write to L2 and L3 when store buffer is drained.<br>Write miss: Put in store buffer, data request for word/line to L3 if not full when store buffer is drained, allocation to L2, write to L3. |

**Table 2-4 Cache operation descriptions (continued)**

| AXI transaction | ARMv6 and ARMv7 memory type attribute | Cache controller behavior |
|---|---|---|
| Cacheable write-back, allocate on write | | Read hit: Read from L2.<br>Read miss: Not cached in L2, causes L3 access.<br>Write hit: Put in store buffer, write to the L2 when store buffer is drained, mark line as dirty.<br>Write miss: Put in store buffer, data request for word/line to L3 if line not full, allocation to L2 when store buffer is drained. |
| Cacheable write-through, allocate on read and write | Outer write-through, allocate on both reads and writes | Read hit: Read from L2.<br>Read miss: Linefill to L2.<br>Write hit: Put in store buffer, write to L2 and L3 when store buffer is drained.<br>Write miss: Put in store buffer, data request for word/line to L3 if line not full, allocation to L2 when store buffer is drained, write to L3. |
| Cacheable write-back, allocate on read and write | Outer write-back, write allocate | Read hit: Read from L2.<br>Read miss: Linefill to L2.<br>Write hit: Put in store buffer, write to L2 when store buffer is drained, mark line as dirty.<br>Write miss: Put in store buffer, data request for word/line to L3 if line not full, allocation to L2 when store buffer is drained. |

**Note**

You can modify the default behavior described in Table 2-4 on page 2-5 using various parameters, such as shared attribute, force write allocate, and exclusive cache configuration.

Other behaviors are described in:

- *Shared attribute*
- *Force write allocate* on page 2-7
- *Exclusive cache configuration* on page 2-7.

### 2.3.1 Shared attribute

The **ARUSERSx[0]** and **AWUSERSx[0]** signals affect transactions. Typically, these signals are driven by ARM processors and reflect the shared attribute as defined in the ARM v6 and v7 architecture. See the shared attribute override enable bit in *Register 1,*

*Auxiliary Control Register* on page 3-11. Shared applies to Normal Memory outer non-cacheable transactions, where **ARCACHESx** or **AWCACHESx** = 0010 or 0011, and transforms them into:

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

**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.

### 2.3.2 Force write allocate

The default setting for the Force write allocate bits in the Auxiliary Control Register is 00, and this configures the cache controller to use the received **AWCACHE** or **ARCACHE** attributes. Additional reference data on the general behavior can be found in Table 2-3 on page 2-4 and Table 2-4 on page 2-5.

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

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

**Note**

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

### 2.3.3 Exclusive cache configuration

Setting the exclusive cache configuration bit in the Auxiliary Control Register to 1 configures the L2 cache to behave as an exclusive cache relative to the L1 cache. The exclusive cache mechanism only applies to the outer write-back inner write-back data transactions received by the cache controller slave ports, that is, **ARCACHESx/AWCACHESx = ARUSER[4:1]/AWUSER[4:1]** = 1011 or 0111 or 1111.

### Reads

For reads, the behavior is as follows:

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

### Writes

For writes, the behavior depends on the value of **AWUSERSx[9:8]**:

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

## 2.4 AXI locked and exclusive accesses

These are as follows:

**AXI locked transfers**

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

In the case of cacheable transfers, 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.

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

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

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

**Note**

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

**AXI exclusive accesses**

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

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

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

**Note**

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

## 2.5 Master and slave port IDs

The AXI ID width on the slave and master ports depend on the value of a parameter, `pl310_AXI_ID_MAX`, which has a default value of 5. The AXI ID width is `[`pl310_AXI_ID_MAX:0]` on the slave ports and `[`pl310_AXI_ID_MAX+2:0]` on the master ports. In this section, it is assumed that 5 is used as the default value for `pl310_AXI_ID_MAX`.

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**
- Master **M0**: 8 bits [7:0], signals **AWIDM0**, **ARIDM0**, **WIDM0**, **BIDM0**, and **RIDM0**
- Master **M1**: 8 bits [7:0], signals **AWIDM1**, **ARIDM1**, **WIDM1**, **BIDM1**, and **RIDM1**.

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

**Table 2-5 Master port ID values for writes**

| Master ID buses | Access types | ID value (Verilog) |
|---|---|---|
| **AWIDMx**, **WIDMx**, **BIDMx** | Non-bufferable write from slave 0 | {AWIDS0, 00} |
| | Non-bufferable write from slave 1 | {AWIDS1, 10} |
| | Eviction from slot 0 | {{`pl310_AXI_ID_MAX-1{0}}, 0011} |
| | Eviction from slot 1 | {{`pl310_AXI_ID_MAX-1{0}}, 0111} |
| | Eviction from slot 2 | {{`pl310_AXI_ID_MAX-1{0}}, 1011} |
| | Write from store buffer slot 0 | {{`pl310_AXI_ID_MAX-1{0}}, 0001} |
| | Write from store buffer slot 1 | {{`pl310_AXI_ID_MAX-1{0}}, 0101} |
| | Write from store buffer slot 2 | {{`pl310_AXI_ID_MAX-1{0}}, 1001} |

Table 2-6 shows the format of the identifications that are exported on the master ports for reads.

**Table 2-6 Master port ID values for reads**

| Master ID buses | Access types | ID value (Verilog) |
|---|---|---|
| **ARIDMx**, **RIDMx** | Non-cacheable or linefill from slave 0 | {ARIDS0, 00} |
| | Non-cacheable or linefill from slave 1 | {ARIDS1, 10} |
| | Read from store buffer slot 0 | {{`pl310_AXI_ID_MAX-1{0}}, 0101} |
| | Read from store buffer slot 1 | {{`pl310_AXI_ID_MAX-1{0}}, 1001} |
| | Read from store buffer slot 2 | {{`pl310_AXI_ID_MAX-1{0}}, 1101} |
| | Prefetch from linefill buffer slot 0 | {{`pl310_AXI_ID_MAX-1{0}}, 0011} |
| | Prefetch from linefill buffer slot 1 | {{`pl310_AXI_ID_MAX-1{0}}, 0111} |
| | Prefetch from linefill buffer slot 2 | {{`pl310_AXI_ID_MAX-1{0}}, 1011} |
| | Prefetch from linefill buffer slot 3 | {{`pl310_AXI_ID_MAX-1{0}}, 1111} |

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.

## 2.6 Exported AXI control

Table 2-7 provides information on the AXI control signals that are exported on the master ports of the cache controller.

**Table 2-7 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 AWPROTMx/ARPROTMx AWLOCKMx/ARLOCKMx** | All as original transaction |
| Linefills associated with a cacheable read transaction | **ARCACHEMx** | As original transaction |
| | **ARPROTMx** | As original transaction |
| | **ARLOCKMx** | {00} |
| Read or write from store buffer, including write-through | **ARCACHEMx/AWCACHEMx** | {0010} |
| | **ARPROTMx/AWPROTMx** | {0,SECURITY, 1} |
| | **ARLOCKMx/AWLOCKMx** | {00} |
| Evictions | **AWCACHEMx** | {1111} |
| | **AWPROTMx** | {0,SECURITY,1} |
| | **AWLOCKMx** | {00} |

**Note**

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

## 2.7 TrustZone support in the cache controller

Some aspects of TrustZone support for the cache controller are:

- 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, any security error is propagated to the processor, and the line is not allocated in L2. The security error generated is dependent on the security mechanism employed by the L3 memory. This can enable secure transfers to access NS memory.
- You can only write to the L2 Control Register with an access tagged as secure, to enable or disable the L2 cache.
- You can only write to the Auxiliary Control Register with an access tagged as secure.
- NS maintenance operations do not clean or invalidate secure data.
- Bit [26] in the Auxiliary Control Register is for NS Lockdown enable and can only be modified with secure accesses. The bit is used to determine whether a lockdown register can be modified by NS accesses.

## 2.8 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:

- *Run mode*
- *Standby mode*
- *Dormant mode*
- *Shutdown mode* on page 2-15.

### 2.8.1 Run mode

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

### 2.8.2 Standby mode

Standby mode consists of externally disabling the clock of the device, while keeping the design powered up. This reduces the power drawn to the static leakage current, plus a small amount of clock power overhead is required to wake up the device from standby mode.

Ensure that you put the cache controller into standby mode when the L1 masters have been put into standby wait for interrupt mode. As soon as the L1 masters wake up and send their first accesses to the cache controller, the L2 cache controller clock must be restarted. You must ensure that the cache controller returns to run mode prior to returning active clock edges to the L1 masters.

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. The cache controller can be put into standby mode when its **IDLE** output signal is asserted.

### 2.8.3 Dormant mode

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

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

1. Save all cache controller configuration registers.

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

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

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

**Note**

These cells are not part of the cache controller.

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

### 2.8.4 Shutdown mode

Shutdown mode powers down the entire device. 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.

## 2.9 Implementation details

This section describes:

- *Disabled operation*
- *Store buffer operation*
- *Cache configurability* on page 2-17
- *Hazards* on page 2-18
- *External error support for L3 memory* on page 2-19
- *Cache event monitoring* on page 2-20
- *Cache interrupts outputs* on page 2-21
- *Parity and RAM error support* on page 2-22
- *Error signalling and handling* on page 2-23.

### 2.9.1 Disabled operation

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

### 2.9.2 Store buffer operation

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

The store buffer has merging capabilities, which means that successive writes to a same line address are merged in the same buffer slot. This means that the slots are not drained as soon as they contain data but wait for potential other accesses targeting the same cache line. The store buffer draining policy is:

- Store buffer slots are drained as soon as they are full.
- Store buffer is drained at each noncacheable read occurrence.
- Store buffer is drained at each nonbufferable write occurrence.
- If the three slots of the store buffer contain data, the least recently accessed is drained.
- If a hazard is detected with one store buffer slot, it is drained to resolve the hazard
- Store buffer slots are drained when a locked transaction is received by one slave port.
- Store buffer slots are drained when a transaction targeting the configuration registers is received by one slave port.

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

The store buffer has three 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.

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

### 2.9.3 Cache configurability

Table 2-8 shows how the cache controller can be configured.

**Table 2-8 Cache controller cache configurability**

| Feature | Enabled by | Range of options | Default value | Default option |
|---|---|---|---|---|
| Cache way size | Register or **WAYSIZE[2:0]** input | 16KB, 32KB, 64KB, 128KB, 256KB, 512KB | 001 | 16KB |
| Number of cache ways | Register or **ASSOCIATIVITY** input | 8, 16[a] | 0 | 8 ways |
| Latency | Register | 1, 2, 3, 4, 5, 6, 7, 8 | 111 | 8 cycles of latency |
| Removal of slave port 1 | Verilog `define pl310_S1 | Commented or uncommented | Commented | No slave port 1 |
| Removal of master port 1 | Verilog `define pl310_M1 | Commented or uncommented | Commented | No master port 1 |
| Parity logic | Verilog `define pl310_PARITY | Commented or uncommented | Commented | No parity logic |

**Table 2-8 Cache controller cache configurability (continued)**

| Feature | Enabled by | Range of options | Default value | Default option |
|---|---|---|---|---|
| Lockdown by master[b] | Verilog `define pl310_LOCKDOWN_BY_MASTER | Commented or uncommented | Commented | No lockdown by master |
| Lockdown by line[b] | Verilog `define pl310_LOCKDOWN_BY_LINE | Commented or uncommented | Commented | No lockdown by line |
| AXI ID width on slave ports | Verilog `define pl310_AXI_ID_MAX <value> | >=1 | 5 | 6 AXI ID bits on slave ports and 8 on master ports |

a. 16-way associativity must be enabled using the pl310_16_WAYS verilog `define.
b. If this feature is selected, you must specify if it is exclusive.

**Note**

If a single slave port, **AXI S1**, is configured it is expected that only a single master port, **AXI M1**, is configured.

### 2.9.4 Hazards

If a hazard is sent by the L1 masters across read and write channels of slave ports, it can result in unpredictable behavior, as described in the *AMBA AXI Protocol*. Hazards can occur when data is present in the cache system, that is, one of the buffers, but not yet present in the cache RAM or L3. The hazard checking in the cache controller is only performed on 16 address bits, bits [20:5].

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

The following hazards can be managed by the cache controller:

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

### 2.9.5 External error support for L3 memory

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

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

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

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

External error support is as follows:

**Write transactions**

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

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

**Read transactions**

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

**Evictions or store buffer drain**

If the request came from an eviction or store buffer 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 2-9 shows error responses for all combinations of L3 access.

**Table 2-9 Error responses for all combinations of L3 access**

| Access type | Error response mechanism |
|---|---|
| Noncacheable read transactions and non-bufferable write transactions | Error response is precise and is passed back to the 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. |
| Evictions and store buffer drains | All error responses are imprecise and are passed back to the L1 processor using the interrupt lines. |

**Note**

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

### 2.9.6 Cache event monitoring

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

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

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

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

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

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

The event monitoring bus is enabled by writing to the event monitoring bus enable bit in the Auxiliary Control Register. Table 2-10 shows the event pins.

**Table 2-10 Event pins**

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

### 2.9.7 Cache interrupts outputs

Table 2-11 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 2-11 Interrupts**

| Pin | Description |
|---|---|
| **DECERRINTR** | Decode error received on master ports from L3 |
| **SLVERRINTR** | Slave error received on master ports from L3 |
| **ERRRDINTR** | Error on L2 data RAM read |

**Table 2-11 Interrupts (continued)**

| Pin | Description |
|---|---|
| **ERRRTINTR** | Error on L2 tag RAM read |
| **ERRWDINTR** | Error on L2 data RAM write |
| **ERRWTINTR** | Error on L2 data RAM write |
| **PARRDINTR** | Parity error on L2 data RAM read |
| **PARRTINTR** | Parity error on L2 tag RAM read |
| **ECNTRINTR** | Event Counter Overflow/Increment |
| **L2CCINTR** | L2CC Combined Interrupt Output |

See *Register 2, Interrupt Registers* on page 3-17 for more information about these outputs.

### 2.9.8 Parity and RAM error support

Figure 2-2 on page 2-23 shows cache controller parity and RAM error support. The cache controller generates the parity write data for the data and tag RAMs. For:

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

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

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

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

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

If a parity error occurs on tag or data RAM during AXI write transactions, a SLVERR 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 2-2 shows the cache controller parity and RAM error support.

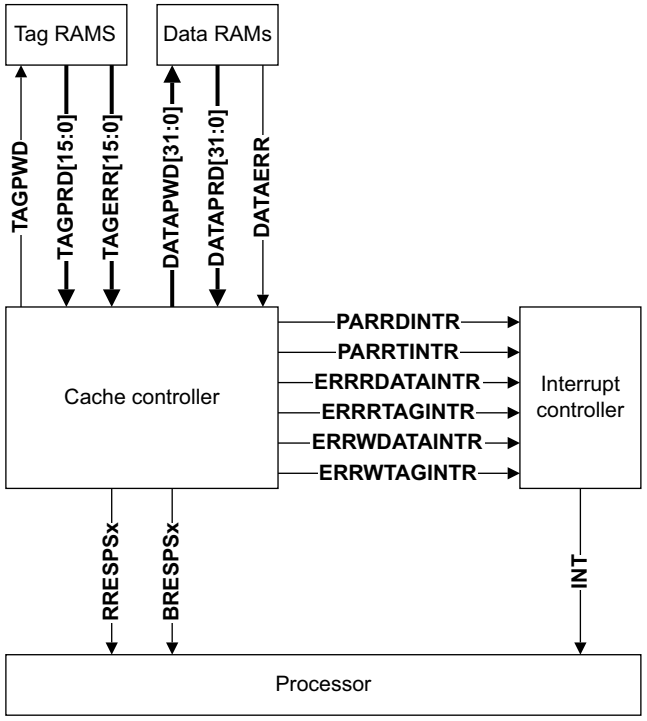



**Figure 2-2 Parity and RAM error support for a 16-way implementation**

### 2.9.9 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 2-24
- *Write access from the store buffer* on page 2-24
- *AXI master or store buffer allocation requests* on page 2-24
- *Clean maintenance* on page 2-25
- *Invalidate maintenance operation* on page 2-25
- *Clean and Invalidate maintenance operation* on page 2-26.

### 2.9.10 Cacheable read requests on AXI slave ports

Table 2-12 shows the error signalling cases and effects when there are cacheable read requests on the AXI salve ports.

**Table 2-12 Cacheable read requests on AXI slave ports**

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

### 2.9.11 Write access from the store buffer

Table 2-13 shows the error signalling cases and effects when there are write-through/write-back write accesses from the store buffer.

**Table 2-13 Write-through/write-back write access from store buffer**

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

### 2.9.12 AXI master or store buffer allocation requests

Table 2-14 shows the error signalling cases and effects when there are AXI master or write-allocate buffer allocation requests.

**Table 2-14 AXI M0 and AXI M1 masters or store buffer allocation requests**

| Case | Effect |
|---|---|
| Tag parity or RAM error on tag read, for next victim selection | An error is flagged through the **PARRTINTR/ERRTINTR** interrupt signals. The allocation is not done. |

**Table 2-14 AXI M0 and AXI M1 masters or store buffer allocation requests (continued)**

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

### 2.9.13 Clean maintenance

Table 2-15 shows the clean maintenance operation error signalling cases and effects.

**Table 2-15 Clean maintenance operation cases**

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

### 2.9.14 Invalidate maintenance operation

Table 2-16 shows the Invalidate maintenance operation error signalling cases and effects.

**Table 2-16 Invalidate maintenance operation cases**

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

### 2.9.15 Clean and Invalidate maintenance operation

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

**Table 2-17 Clean and Invalidate maintenance operation cases**

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

## 2.10 MBIST support

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

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

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

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

Figure 2-3 on page 2-28 shows the cache controller MBIST interface.

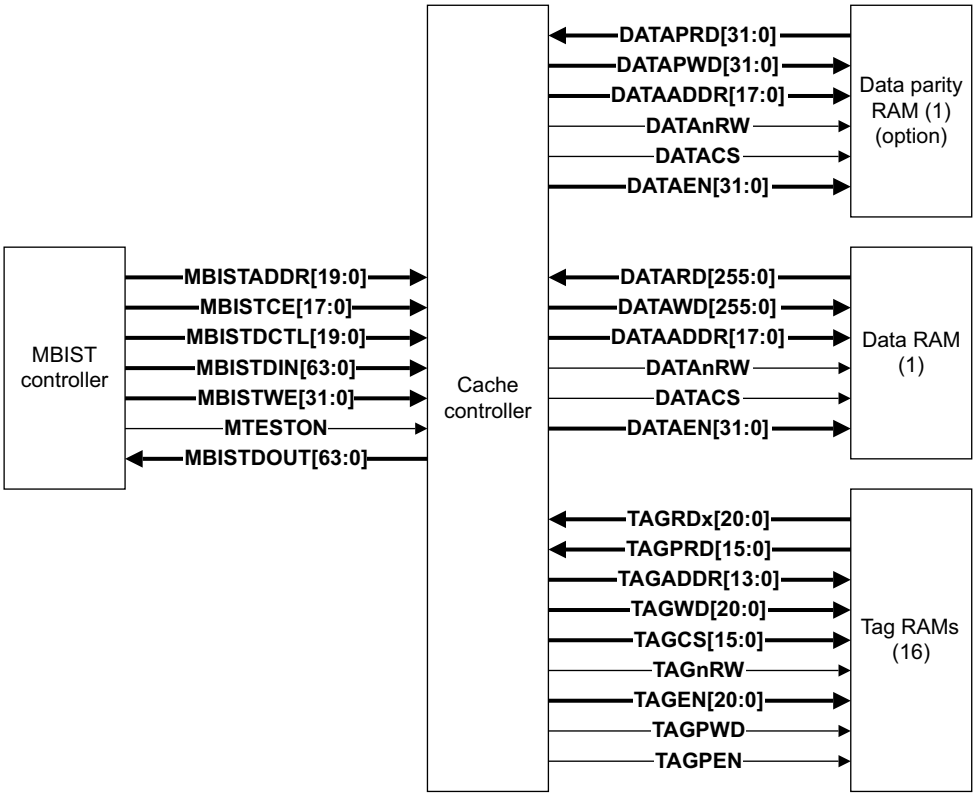



**Figure 2-3 MBIST interface for 16-way implementation, with parity, without lockdown by line**

**Note**

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

## 2.11 RAM organization

This section describes the following:

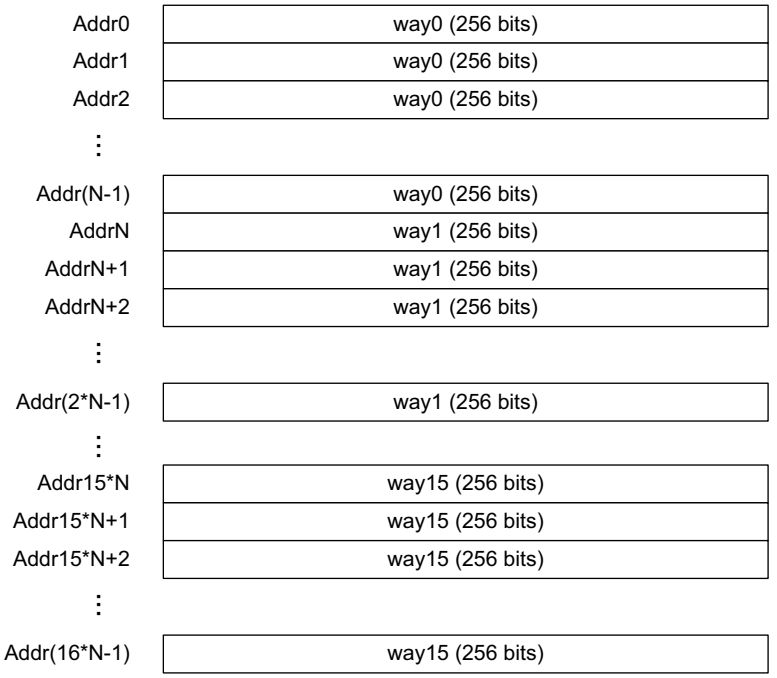
- *Data RAM*
- *Tag RAM* on page 2-30
- *Parity RAM* on page 2-32
- *RAM bus usage versus cache associativity and way size* on page 2-32.

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

### 2.11.1 Data RAM

The data RAM shown in Figure 2-4 on page 2-30 is organized as *n*-way 256-bit wide contiguous memories, where *n* is 8 or 16. It supports the following accesses:

- 8 word data reads
- x * 8 bits data writes with byte enables controls
- 8 word data writes for line allocations.




| L2 Cache Size | N = |
|---|---|
| 256KB | 512 |
| 512KB | 1024 |
| 1MB | 2048 |
| 2MB | 4096 |
| 4MB | 8192 |
| 8MB | 16386 |

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

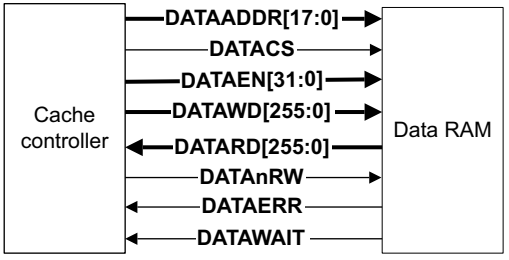



**Figure 2-4 Data RAM organization for 16 ways**

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

### 2.11.2 Tag RAM

The tag RAM is shown in Figure 2-5 on page 2-31. There is one tag RAM for each way of the L2 cache. A tag RAM is organized as a 23-bit, 22-bit, or 21-bit wide memory:

- 18 bits for address tag

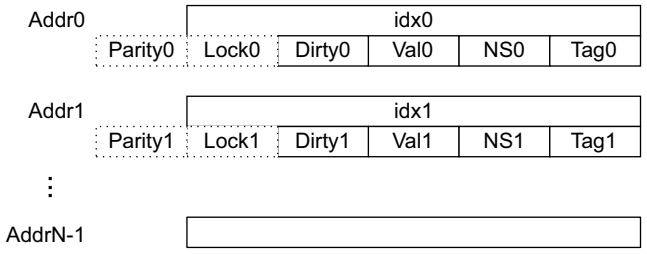
  

- 1 bit for security information
- 1 bit for valid information
- 1 bit for dirty information
- 1 optional bit for lock information.
- 1 optional bit for parity.

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

**Note**

You require a 23-bit wide memory to support both parity and lockdown by line option. You require a 22-bit wide memory to support either the parity option or the lockdown by line option.




| L2 Cache Size | N = |
|---|---|
| 256KB | 512 |
| 512KB | 1024 |
| 1MB | 2048 |
| 2MB | 4096 |
| 4MB | 8192 |
| 8MB | 16386 |

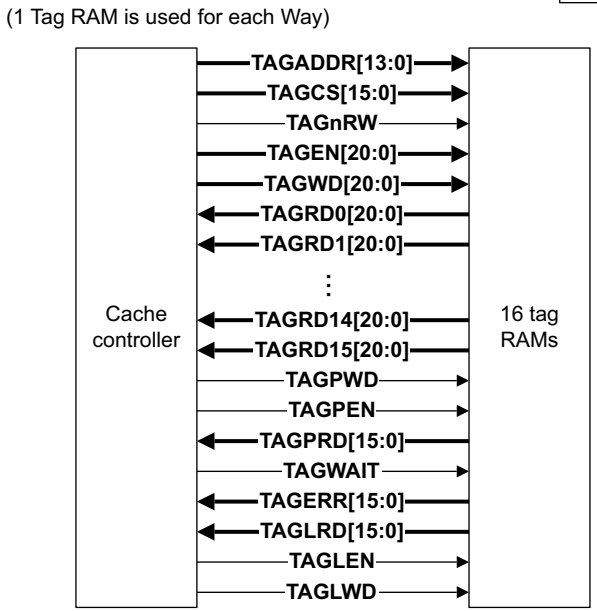



**Figure 2-5 Tag RAM organization for 16 ways, with parity, without lockdown by line**

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

### 2.11.3 Parity RAM

Optional parity RAM is described in:

- *Tag parity RAM*
- *Data parity RAM*.

#### Tag parity RAM

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

#### Data parity RAM

Figure 2-6 shows the required RAM organization for data parity RAM.

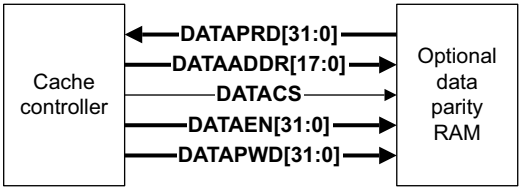



**Figure 2-6 Data parity RAM organization**

Data parity RAM is always 32 bits wide, and connection is the same as for data RAM described in *Data RAM* on page 2-29.

### 2.11.4 RAM bus usage versus cache associativity and way size

This section describes:

- Data RAM usage
- Tag RAM usage.

**Data RAM usage**

Figure 2-7 on page 2-33 shows the **DATAADDR** bus format:

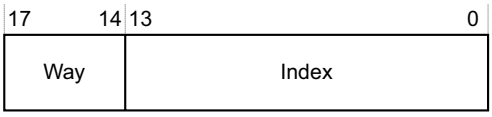



**Figure 2-7 Data RAM address bus format for 16 ways**

**Note**

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

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

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

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

### Tag RAM usage

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

- 16KB way size:
  - Bits [13:9] of **TAGADDR** are left unconnected.

  - Bits [8:0] are the RAM address bus. All bits of **TAGRDn** and **TAGWDn** are used for data buses.
- 32KB way size:
  - Bits [13:10] of **TAGADDR** are left unconnected.
  - Bits [9:0] are the RAM address bus.
  - Bits [20:1] of **TAGRDn** and **TAGWDn** are used for data buses.
  - Bit [0] of **TAGRDn** must be tied LOW.
- 64KB way size:
  - Bits [13:11] of **TAGADDR** are left unconnected.
  - Bits [10:0] are the RAM address bus.
  - Bits [20:2] of **TAGRDn** and **TAGWDn** are used for data buses.
  - Bits [1:0] of **TAGRDn** must be tied LOW.
- 128KB way size:
  - Bits [13:12] of **TAGADDR** are left unconnected.
  - Bits [11:0] are the RAM address bus.
  - Bits [20:3] of **TAGRDn** and **TAGWDn** are used for data buses.
  - Bits [2:0] of **TAGRDn** must be tied LOW.
- 256KB way size:
  - Bit [13] of **TAGADDR** is left unconnected.
  - Bits [20:4] of **TAGRDn** and **TAGWDn** are used for data buses.
  - Bits [3:0] of **TAGRDn** must be tied LOW.
- 512KB way size:
  - All bits [13:0] are the RAM address bus.
  - Bits [20:5] of **TAGRDn** and **TAGWDn** are used for data buses.
  - Bits [4:0] of **TAGRDn** must be tied LOW.

# Chapter 3
# **Programmer's Model**

This chapter describes the 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.

  The cache controller is controlled through a set of memory-mapped registers that occupy a re-locatable 4KB of memory. You can access the registers through direct address decoding in the slave ports. The base address for these ports is provided by **REGFILEBASE[31:12]**.

- 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.

When accessing the registers:

- Bits [1:0] of the address must be zero, otherwise a SLVERR response is returned.
- The burst length must be equal to zero, otherwise a SLVERR response is returned.
- Only 32-bit accesses are allowed, otherwise a SLVERR response is returned.
- Exclusive accesses are not allowed. A SLVERR response is returned.

### 3.1.1 Initialization sequence

**Caution**

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 `0xFFFF` to `0x77C`
   - Poll cache maintenance register until invalidate operation is complete.
3. Write to the Lockdown D and Lockdown I Register 9 if required.
4. Write to interrupt clear register to clear any residual raw interrupts set.
5. Write to Control Register 1 with the LSB set to 1 to enable the cache.

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 the Auxiliary Control Register.

**Note**

All unmapped registers are read as zero. Writes to an unmapped register have no effect and return a OKAY response.

## 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 | Cache ID and Cache Type | Ignored | NS/S |
| 1 | Control | Control | Write S<br>Read NS/S |
| 2 | Interrupt/Counter Control Registers | Interrupt/Counter Control Registers | NS/S |
| 3-6 | Reserved | Reserved | - |
| 7 | Cache Maintenance Operations | Cache Maintenance Operations | Secure bit of access affects operation |
| 8 | Reserved | Reserved | - |
| 9 | Cache Lockdown | Cache Lockdown | Secure bit of access affects operation |
| 10-11 | Reserved | Reserved | - |
| 12 | Address Filtering | Address Filtering | Write S<br>Read NS/S |
| 13-14 | Reserved | Reserved | - |
| 15 | Debug | Debug | Write S<br>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 | 0x410000C0[a] | *Register 0, Cache ID Register* on page 3-8 |
| r0 | Cache Type | 0x004 | RO | 0x1C100100[b] | *Register 0, Cache Type Register* on page 3-9 |
| r1 | Control | 0x100 | RW | 0x00000000 | *Register 1, Control Register* on page 3-10 |

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

| Register | Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|---|
| r1 | Auxiliary Control | 0x104 | RW | 0x02020FFF[b] | *Register 1, Auxiliary Control Register* on page 3-11 |
| r2 | Event Counter Control | 0x200 | RW | 0x00000000 | *Register 2, Event Counter Control Register* on page 3-14 |
| r2 | Event Counter1 Configuration | 0x204 | RW | 0x00000000 | *Register 2, Event Counter Configuration Registers* on page 3-15 |
| r2 | Event Counter0 Configuration | 0x208 | RW | 0x00000000 | |
| r2 | Event Counter1 Value | 0x20C | RW | 0x00000000 | *Register 2, Event Counter Value Registers* on page 3-16 |
| r2 | Event Counter0 Value | 0x210 | RW | 0x00000000 | |
| r2 | Interrupt Mask[c] | 0x214 | RW | 0x00000000 | *Register 2, Interrupt Registers* on page 3-17 |
| r2 | Masked Interrupt Status[c] | 0x218 | RO | 0x00000000 | |
| r2 | Raw Interrupt Status[c] | 0x21C | RO | 0x00000000 | |
| r2 | Interrupt Clear[c] | 0x220 | WO | 0x00000000 | |
| r7 | Cache Sync | 0x730 | RW | 0x00000000 | *Register 7, Cache Maintenance Operations* on page 3-21 |
| r7 | Invalidate Line By PA | 0x770 | RW | 0x00000000 | |
| r7 | Invalidate by Way | 0x77C | RW | 0x00000000 | |
| r7 | Clean Line by PA | 0x7B0 | RW | 0x00000000 | |
| r7 | Clean Line by Index/Way | 0x7B8 | RW | 0x00000000 | |
| r7 | Clean by Way | 0x7BC | RW | 0x00000000 | |
| r7 | Clean and Invalidate Line by PA | 0x7F0 | RW | 0x00000000 | |
| r7 | Clean and Invalidate Line by Index/Way | 0x7F8 | RW | 0x00000000 | |
| r7 | Clean and Invalidate by Way | 0x7FC | RW | 0x00000000 | |
| r9 | Data Lockdown 0 by Way | 0x900 | RW | 0x00000000 | *Register 9, Cache Lockdown* on page 3-25 |

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

| Register | Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|---|
| r9 | Instruction Lockdown 0 by Way | 0x904 | RW | 0x00000000 | *Register 9, Cache Lockdown* on page 3-25 |
| r9 | Data Lockdown 1 by Way[d] | 0x908 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 1 by Way[d] | 0x90C | RW | 0x00000000 | |
| r9 | Data Lockdown 2 by Way[d] | 0x910 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 2 by Way[d] | 0x914 | RW | 0x00000000 | |
| r9 | Data Lockdown 3 by Way[d] | 0x918 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 3 by Way[d] | 0x91C | RW | 0x00000000 | |
| r9 | Data Lockdown 4 by Way[d] | 0x920 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 4 by Way[d] | 0x924 | RW | 0x00000000 | |
| r9 | Data Lockdown 5 by Way[d] | 0x928 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 5 by Way[d] | 0x92C | RW | 0x00000000 | |
| r9 | Data Lockdown 6 by Way[d] | 0x930 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 6 by Way[d] | 0x934 | RW | 0x00000000 | |
| r9 | Data Lockdown 7 by Way[d] | 0x938 | RW | 0x00000000 | |
| r9 | Instruction Lockdown 7 by Way[d] | 0x93C | RW | 0x00000000 | |
| r9 | Lockdown by Line Enable[e] | 0x950 | RW | 0x00000000 | |
| r9 | Unlock All Lines by Way[e] | 0x954 | RW | 0x00000000 | |

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

| Register | Name | Base offset | Type | Reset value | Description |
|---|---|---|---|---|---|
| r12 | Address Filtering Start | 0xC00 | RW | 0x00000000 | *Register 12, Address Filtering* on page 3-32 |
| r12 | Address Filtering End | 0xC04 | RW | 0x00000000 | |
| r15 | Debug Control Register | 0xF40 | RW | 0x00000000 | *Register 15, Debug Register* on page 3-33 |

a. This value is pin dependent, depending on how external CACHEID pins are tied.
b. This value is pin dependent, depending on how external WAYSIZE and ASSOCIATIVITY pins are tied.
c. The cache interrupt registers are those that can be accessed by secure and non-secure operations.
d. These registers are implemented if the option `pl310_LOCKDOWN_BY_MASTER` is enabled. Otherwise, they are unused.
e. These registers are implemented if the option `pl310_LOCKDOWN_BY_LINE` is enabled. Otherwise, they are unused.

## 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

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

Figure 3-1 shows the register bit assignments.

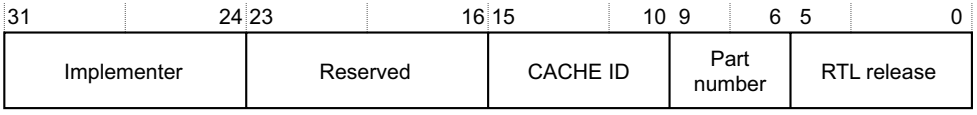



**Figure 3-1 Cache ID Register bit assignments**

Table 3-3 shows the register bit assignments.

**Table 3-3 Cache ID Register bit assignments**

| Bits | Field | Description |
|---|---|---|
| [31:24] | Implementer | `0x41` (ARM) |
| [23:16] | Reserved | SBZ |
| [15:10] | CACHE ID | - |
| [9:6] | Part number | `0x3` |
| [5:0] | RTL release | `0x0` |

**Note**

- Part number `0x3` denotes ARM PL310 Level 2 Cache Controller
- RTL release `0x0` denotes r0p0 code of the cache controller. See the Release Note for 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.

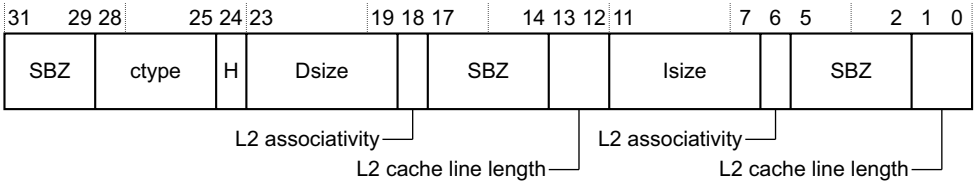



**Figure 3-2 Cache Type Register bit assignments**

Table 3-4 shows the register bit assignments.

**Table 3-4 Cache Type Register bit assignments**

| Bits | Field | Sub-field | Comments |
|---|---|---|---|
| [31:29] | SBZ | | 000 |
| [28:25] | ctype | | 11*xy*, where:<br>*x*=1 if `pl310_LOCKDOWN_BY_MASTER` is defined, otherwise 0<br>*y*=1 if `pl310_LOCKDOWN_BY_LINE` is defined, otherwise 0.<br>See *Register 9, Cache Lockdown* on page 3-25 for information on the lockdown features. |
| [24] | H | | 0 - unified[a] |
| [23:19] | Dsize | | - |
| | [23] | SBZ/RAZ | 0 |
| | [22:20] | L2 cache way size | Read from Auxiliary Control Register [19:17] |
| | [19] | SBZ/RAZ | 0 |
| [18] | L2 associativity | | Read from Auxiliary Control Register [16] |
| [17:14] | SBZ | | 0 |
| [13:12] | L2 cache line length | | 00 - 32 bytes |

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

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

a. 1 = Harvard.

### 3.3.3 Register 1, Control Register

The Control Register is a read and write register. This register enables or disables the cache controller. 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 all outstanding transactions to complete and all buffers to be empty by performing a cache sync.
2. Update register.
3. Return write response.

Figure 3-3 shows the register bit assignments.

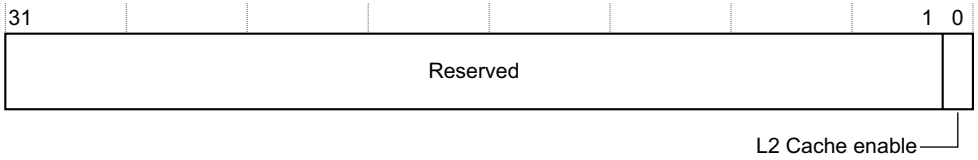



**Figure 3-3 Control Register bit assignments**

Table 3-5 shows the register bit assignments.

**Table 3-5 Control Register bit assignments**

| Bits | Field | Description |
|---|---|---|
| [31:1] | Reserved | SBZ/RAZ |
| [0] | L2 Cache enable | 0 = L2 Cache is disabled. This is the default value.<br>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 on page 3-12 shows the register bit assignments.

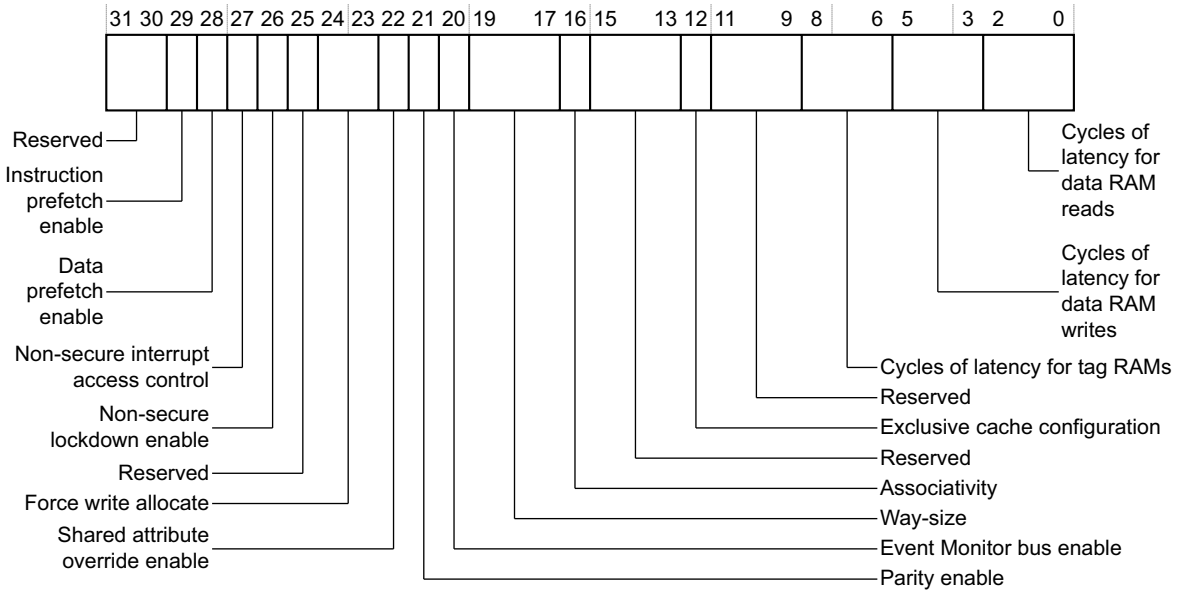



**Figure 3-4 Auxiliary Control Register bit assignments**

Table 3-6 shows the register bit assignments.

**Table 3-6 Auxiliary Control Register bit assignments**

| Bits | Field | Description |
|---|---|---|
| [31:30] | Reserved | SBZ/RAZ |
| [29] | Instruction prefetch enable[a] | 0 = Instruction prefetching disabled. This is the default.<br>1= Instruction prefetching enabled. |
| [28] | Data prefetch enable[a] | 0 = Data prefetching disabled. This is the default.<br>1= Data prefetching enabled. |
| [27] | Non-secure interrupt access control | 0 = Interrupt Clear (`0x220`) and Interrupt Mask (`0x214`) can only be modified or read with secure accesses. This is the default.<br>1 = Interrupt Clear (`0x220`) and Interrupt Mask (`0x214`) can be modified or read with secure or non-secure accesses. |
| [26] | Non-secure lockdown enable | 0 = Lockdown registers cannot be modified using non-secure accesses. This is the default.<br>1 = Non-secure accesses can write to the lockdown registers. |
| [25] | Reserved | SBO/RAO |

**Table 3-6 Auxiliary Control Register bit assignments (continued)**

| Bits | Field | Description |
|---|---|---|
| [24:23] | Force write allocate | 00 = Use **AWCACHE** attributes for WA. This is the default.<br>01 = Force no allocate, set WA bit always 0.<br>10 = Override **AWCACHE** attributes, set WA bit always 1 (all cacheable write misses become write allocated).<br>11 = Internally mapped to 00. See *Cache operation* on page 2-5 for more details. |
| [22] | Shared attribute override enable | 0 = Treats shared accesses as specified in *Shared attribute* on page 2-6. This is the default.<br>1 = Shared attribute internally ignored. |
| [21] | Parity enable | 0 = Disabled. This is the default.<br>1 = Enabled. |
| [20] | Event monitor bus enable | 0 = Disabled. This is the default.<br>1 = Enabled. |
| [19:17] | Way-size | 000 = Reserved, internally mapped to 16KB.<br>001 = 16KB. This is the default.<br>010 = 32KB.<br>011 = 64KB.<br>100 = 128KB.<br>101 = 256KB.<br>110 = 512KB.<br>111 = Reserved, internally mapped to 512 KB. |
| [16:13] | Associativity | 0 = 8-way. This is the default.<br>1 = 16-way. |
| [15:13] | Reserved | SBZ/RAZ |
| [12] | Exclusive cache configuration | 0 = Disabled. This is the default.<br>1 = Enabled, see *Exclusive cache configuration* on page 2-7. |
| [11:9] | Reserved | SBZ/RAZ |

**Table 3-6 Auxiliary Control Register bit assignments (continued)**

| Bits | Field | Description |
|---|---|---|
| [8:6] | Cycles of latency for tag RAM | 000 = 1 cycle of latency, there is no additional latency.<br>001 = 2 cycles of latency.<br>010 = 3 cycles of latency.<br>011 = 4 cycles of latency.<br>100 = 5 cycles of latency.<br>101 = 6 cycles of latency.<br>110 = 7 cycles of latency.<br>111 = 8 cycles of latency. This is the default.<br>See *Cache controller compiled RAM latencies* for more information. |
| [5:3] | Cycles of latency for data RAM writes | |
| [2:0] | Cycles of latency for data RAM reads | |

a. When enabled, as soon as a cacheable read transaction is received by one of the slave ports, a cache lookup is done on the sequentially following cache line. In case of miss, the cache line is prefetched from L3 and allocated into the L2 cache. The prefetch mechanism is not launched in the case of 4KB boundary crossing.

### 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 shows a compiled RAM latency.

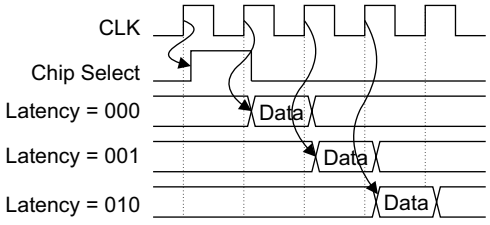



**Figure 3-5 Compiled RAM latency examples for reads**

## 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 non-secure operations. Figure 3-6 on page 3-15 shows the register bit assignments.

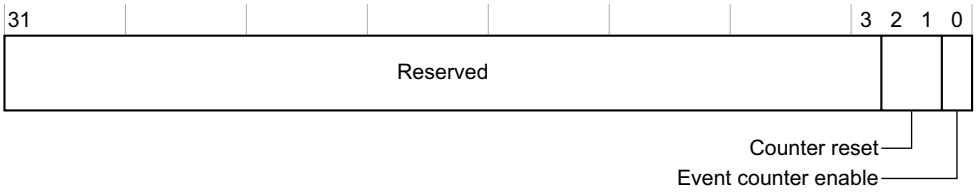



**Figure 3-6 Event Counter Control Register bit assignments**

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:<br>Bit 2 = Event Counter1 reset<br>Bit 1 = Event Counter0 reset. |
| [0] | Event counter enable | 0 = Event Counting Disable. This is the default.<br>1 = Event Counting Enable. |

### 3.3.6 Register 2, Event Counter Configuration Registers

The Event Counter Configuration Registers are read and write registers. These registers enable event counter 1 and 0 to be driven by a specific event. When the event occurs it causes the counter1 or 0 to increment. The counter event source signals are described in *Cache event monitoring* on page 2-20. This register can be accessed by secure and NS operations. Figure 3-7 shows the register bit assignments.

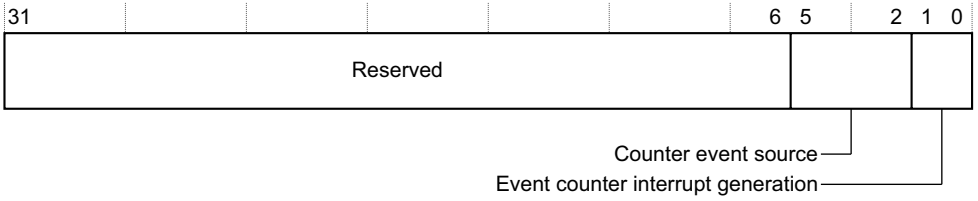



**Figure 3-7 Event Counter Configuration Register bit assignments**

Table 3-8 shows the register bit assignments.

**Table 3-8 Event Counter Configuration Register bit assignments**

| Bits | Field | Description | |
|---|---|---|---|
| [31:6] | Reserved | SBZ/RAZ | |
| [5:2] | Counter event source | Event | Encoding |
| | | Counter Disabled | 0000 |
| | | **CO** | 0001 |
| | | **DRHIT** | 0010 |
| | | **DRREQ** | 0011 |
| | | **DWHIT** | 0100 |
| | | **DWREQ** | 0101 |
| | | **DWTREQ** | 0110 |
| | | **IRHIT** | 0111 |
| | | **IRREQ** | 1000 |
| | | **WA** | 1001 |
| | | Counter Disabled | 1010-1111 |
| [1:0] | Event counter interrupt generation | 00 = Disabled. This is the default.<br>01 = Enabled: Increment condition.<br>10 = Enabled: Overflow condition.<br>11 = Interrupt generation is disabled. | |

**Note**

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

### 3.3.7 Register 2, Event Counter Value Registers

The Event Counter Value Registers are read and write registers. These registers enable the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration Registers. The counter can be preloaded if counting is

disabled and reset by the Event Counter Control Register. This register can be accessed by secure and non-secure operations. Table 3-9 shows the register bit assignments. These register can only be written to when bits [5:2] of the Event Counter Configuration Registers are set to Counter Disabled.

**Table 3-9 Event Counter 1 Value Register bit assignments**

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

### 3.3.8 Register 2, Interrupt Registers

There are four interrupt registers:

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

Figure 3-8 shows the register bit assignments.

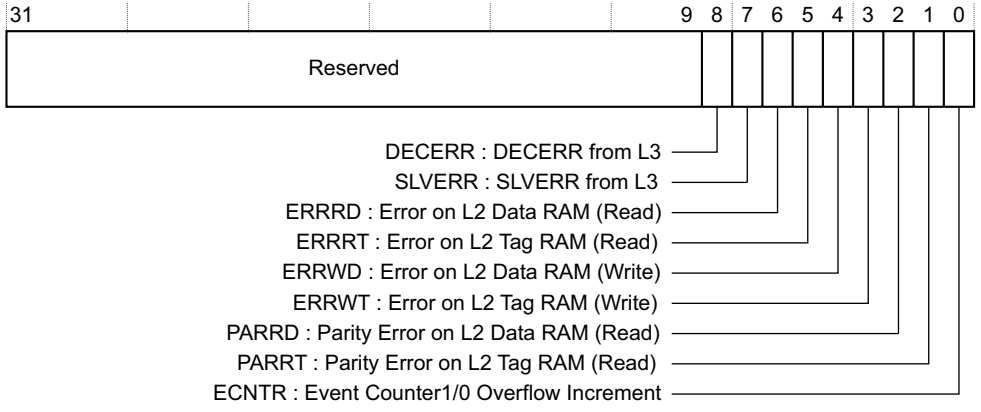



**Figure 3-8 Interrupt Register bit assignments**

### Interrupt Mask Register

The 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. Figure 3-8 on page 3-17 shows the register bit assignments. The bit assignments enables the masking of the interrupts on both their individual outputs and the combined **L2CCINTR** line. Clearing a bit by writing a 0, disables the interrupt triggering on that pin. All bits are cleared by a reset. You must write to the register bits with a 1 to enable the generation of interrupts. Non-secure access to this register is dependent on Auxiliary Control Register bit [27].

Table 3-10 shows the register bit assignments.

**Table 3-10 Interrupt Mask Register bit assignments**

| Bits | Field | Description |
|---|---|---|
| [31:9] | Reserved | SBZ/RAZ |
| [8] | DECERR: DECERR from L3 | 1 = Enable.<br>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 Counter1/0 Overflow Increment | |

### 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. This register can be accessed by secure and non-secure operations. The register gives an AND function of the raw interrupt status with the values of the interrupt mask register. All the bits are cleared by a reset.

Table 3-11 shows the register bit assignments.

**Table 3-11 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:<br>If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.<br>If bits read LOW, either no interrupt has been generated or the interrupt is masked. |
| [7] | SLVERR: SLVERR from L3 | |
| [6] | ERRRD: Error on L2 data RAM (Read) | |
| [5] | ERRRT: Error on L2 tag RAM (Read) | |
| [4] | ERRWD: Error on L2 data RAM (Write) | |
| [3] | ERRWT: Error on L2 tag RAM (Write) | |
| [2] | PARRD: Parity Error on L2 data RAM (Read) | |
| [1] | PARRT: Parity Error on L2 tag RAM (Read) | |
| [0] | ECNTR: Event Counter1/0 Overflow Increment | |

### 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 non-secure operations. All the bits are cleared by a reset.

Table 3-12 shows the register bit assignments.

**Table 3-12 Raw Interrupt Status Register bit assignments**

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

### Interrupt Clear Register

The Interrupt Clear Register is a write-only register and all bits are cleared by a reset. Non-secure access to this register is dependent on Auxiliary Control Register bit [27].

Table 3-13 shows the register bit assignments.

**Table 3-13 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<br>When a bit is written as 0, it has no effect |
| [7] | SLVERR: SLVERR from L3 | |
| [6] | ERRRD: Error on L2 data RAM (Read) | |
| [5] | ERRRT: Error on L2 tag RAM (Read) | |
| [4] | ERRWD: Error on L2 data RAM (Write) | |
| [3] | ERRWT: Error on L2 tag RAM (Write) | |
| [2] | PARRD: Parity Error on L2 data RAM (Read) | |
| [1] | PARRT: Parity Error on L2 tag RAM (Read) | |
| [0] | ECNTR: Event Counter1/0 Overflow Increment | |

## 3.3.9 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, their behavior is presented in the following manner:

**Secure access**

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

**Non-secure access**

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

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

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

Table 3-14 shows the cache maintenance operations. They are executed by writing to the Cache Operations Register 7. See also Table 3-15 on page 3-25.

**Table 3-14 Maintenance operations**

| Operation | Base offset | Type | Bit assignment format |
|---|---|---|---|
| Cache Sync | 0x730 | RW | - |
| Invalidate Line by PA | 0x770 | RW | See Figure 3-9 |
| Invalidate by Way | 0x77C | RW | See Figure 3-11 on page 3-24 |
| Clean Line by PA | 0x7B0 | RW | See Figure 3-9 |
| Clean Line by Index/Way | 0x7B8 | RW | See Figure 3-10 on page 3-23 |
| Clean by Way | 0x7BC | RW | See Figure 3-11 on page 3-24 |
| Clean and Invalidate Line by PA | 0x7F0 | RW | See Figure 3-9 |
| Clean and Invalidate Line by Index/Way | 0x7F8 | RW | See Figure 3-10 on page 3-23 |
| Clean and Invalidate by Way | 0x7FC | RW | See Figure 3-11 on page 3-24 |

Figure 3-9 shows the PA format.

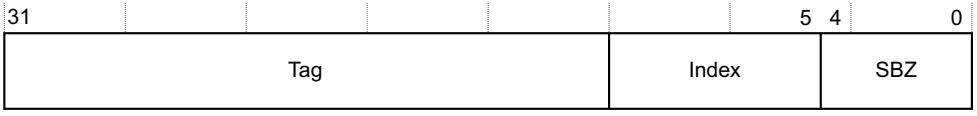



**Figure 3-9 Physical address format**

Figure 3-10 on page 3-23 shows the Index or Way format.

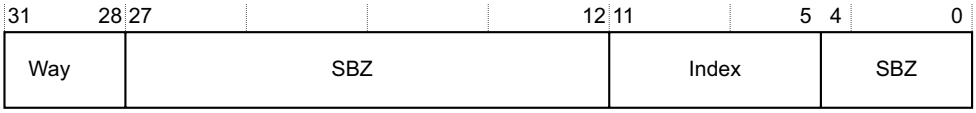



**Figure 3-10 Index/way format**

**Note**

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

### Atomic operations

The following are atomic operations:

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

These operations stall the slave ports until they are complete.

### Background operations

The following operations are run as background tasks:

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

**Note**

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

Writing to the register starts the operation on the Ways set to 1 in bits [15:0]. When a Way bit is set to 1, it is reset to 0 when the corresponding way is totally cleaned or invalidated. You must poll Register 7 to see when the operation is complete, indicated by all bits cleared.

Figure 3-11 on page 3-24 shows the Format C lockdown. You can select multiple ways at the same time, by setting the Way bits to 1.

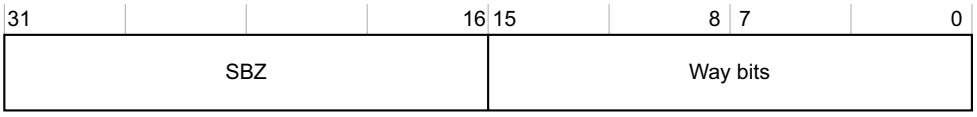



**Figure 3-11 Format C lockdown**

**Note**

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

During background operations, the software must not access these ways. Any write to a configuration and control register while a background operation is running results in 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, that is, data accessed during the clean operation. To ensure that a clean operation is completed, mask the interrupts. Also ensure that the software polls on the Cache Operation Register to check if the operation is complete.

Table 3-15 shows the cache maintenance operations.

**Table 3-15 Cache maintenance operations**

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

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

### 3.3.10 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.

**Note**

Register 7 operations that invalidate, clean, or clean and invalidate cache contents affect locked-down cache lines as normal.

This register has read-only or read and write permission, depending on the security you have selected for the register access and on the Non-Secure Lockdown Enable bit in the Auxiliary Control Register. Table 3-16 shows the different settings of the Cache Lockdown Register.

**Table 3-16 Cache lockdown**

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

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

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

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

### Cache lockdown by line

This feature is optional, and can be enabled using the `pl310_LOCKDOWN_BY_LINE` parameter. You can program a period of time during which all allocated cache lines are marked as locked and are then considered as locked, preventing them from being evicted. The locking period is enabled when bit [0] of the Lockdown by Line Enable Register is set. See Table 3-17.

**Table 3-17 Lockdown by Line Enable Register bit assignments**

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

The locked status of each cache line is given by the optional bit [21] of the Tag RAM. During a locking period, the choice of way for allocations is described in *Cache lockdown by way*.

You can unlock all lines marked as locked by the Lock by Line mechanism, by using the Unlock All Lines operation, which is a background operation. You can check the status of this operation by reading the Unlock All Lines register. See Table 3-18.

While an Unlock All Lines operation is in progress, you cannot launch a background cache maintenance operation. If attempted, a SLVERR is returned.

**Table 3-18 Unlock All Lines Register bit assignments**

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

### Cache lockdown by way

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 allocations. 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 3-11 on page 3-24. 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 16-way set associative cache has 16 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 16 possible locations where the new line can be placed. Lockdown format C restricts the cache replacement algorithm to only use a subset of the 16 possible locations.

If the Lockdown by Master feature is enabled through the `pl310_LOCKDOWN_BY_MASTER` option, you can lock ways differently based on different master IDs, provided by **AR/WUSERSx[7:5]**. If the Lockdown by Master feature is not enabled, you can only define a different lockdown mechanism for data and instructions.

The locking configurability is shown in Table 3-19 to Table 3-34 on page 3-31. To apply lockdown, set each bit to 1 to lock each respective Way. For example, set Bit [0] for Way 0, Bit [1] for Way 1.

**Table 3-19 Data Lockdown 0 Register, offset 0x900**

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

**Table 3-20 Instruction Lockdown 0 Register, offset 0x904**

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

**Table 3-21 Data Lockdown 1 Register, offset 0x908**

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

**Table 3-22 Instruction Lockdown 1 Register, offset 0x90C**

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

**Table 3-23 Data Lockdown 2 Register, offset 0x910**

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

**Table 3-24 Instruction Lockdown 2 Register, offset 0x914**

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

**Table 3-25 Data Lockdown 3 Register, offset 0x918**

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

**Table 3-26 Instruction Lockdown 3 Register, offset 0x91C**

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

**Table 3-27 Data Lockdown 4 Register, offset 0x920**

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

**Table 3-28 Instruction Lockdown 4 Register, offset 0x924**

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

**Table 3-29 Data Lockdown 5 Register, offset 0x928**

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

**Table 3-30 Instruction Lockdown 5 Register, offset 0x92C**

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

**Table 3-31 Data Lockdown 6 Register, offset 0x930**

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

**Table 3-32 Instruction Lockdown 6 Register, offset 0x934**

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

**Table 3-33 Data Lockdown 7 Register, offset 0x938**

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

**Table 3-34 Instruction Lockdown 7 Register, offset 0x93C**

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

**Note**

- If the `pl310_16_WAYS` option is not implemented, bits [15:8] are reserved in all the Data and Instruction Lockdown registers.
- The Data and Instruction Lockdown 1-7 registers are not used if the option `pl310_LOCKDOWN_BY_MASTER` is not enabled.

### Replacement strategy

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

The pseudo-random replacement strategy fills 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.

### 3.3.11 Register 12, Address Filtering

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

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

This feature is programmed using two registers.

#### Address Filtering Start Register

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

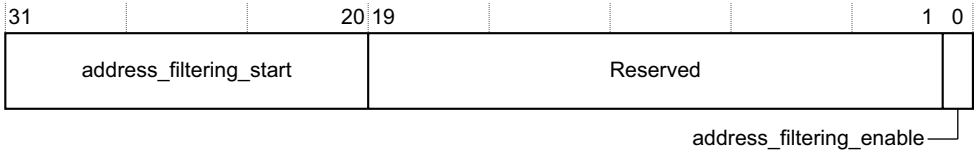



**Figure 3-12 Address Filtering Start Register bit assignments**

Table 3-35 shows the register bit assignments.

**Table 3-35 Address Filtering Start Register bit assignments**

| Bits | Field | Description |
|---|---|---|
| [31:20] | address_filtering_start | Address filtering start address |
| [19:1] | Reserved | SBZ/RAZ |
| [0] | address_filtering_enable | 0 = Address filtering disabled. This is the default.<br>1 = Address filtering enabled. |

**Note**

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

### Address Filtering End Register

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

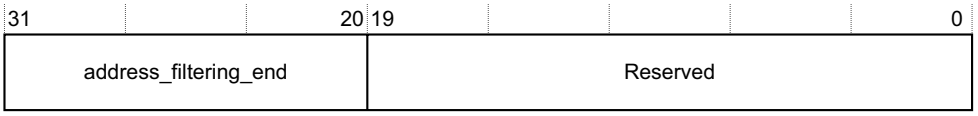



**Figure 3-13 Address Filtering End Register bit assignments**

Table 3-36 shows the register bit assignments.

**Table 3-36 Address Filtering End Register bit assignments**

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

## 3.3.12 Register 15, Debug Register

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

Figure 3-14 shows the register bit assignments.

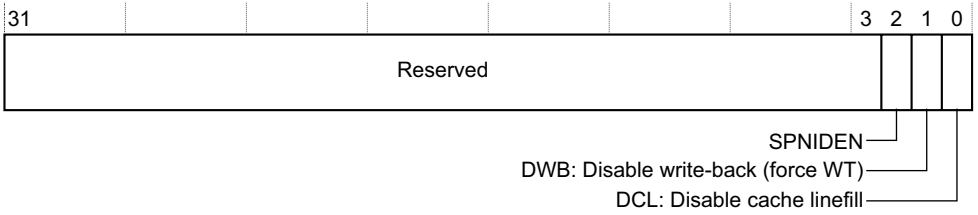



**Figure 3-14 Debug Control Register bit assignments**

Table 3-37 shows the register bit assignments.

**Table 3-37 Debug Control Register bit assignments**

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

**Disabling cache linefills**

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

**Note**

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

# Appendix A
# **Signal Descriptions**

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

## A.1 Clock and reset

Table A-1 shows the clock and reset signals.

**Table A-1 Clock and reset signals**

| Signal | Type | Description |
|---|---|---|
| **CLK** | Input | Main clock |
| **CLKOUT** | Output | Global clock signal for RAMs |
| **IDLE** | Output | Indicates cache controller is idle |
| **INCLKENM0** | Input | Clock enable for M0 AXI inputs |
| **INCLKENM1** | Input | Clock enable for M1 AXI inputs |
| **INCLKENS0** | Input | Clock enable for S0 AXI inputs |
| **INCLKENS1** | Input | Clock enable for S1 AXI inputs |
| **nRESET** | Input | Global reset, active LOW |
| **OUTCLKENM0** | Input | Clock enable for M0 AXI outputs |
| **OUTCLKENM1** | Input | Clock enable for M1 AXI outputs |
| **OUTCLKENS0** | Input | Clock enable for S0 AXI outputs |
| **OUTCLKENS1** | Input | Clock enable for S1 AXI outputs |

## A.2 Configuration

Table A-2 shows the configuration signals.

**Table A-2 Configuration signals**

| Signal | Type | Description |
|---|---|---|
| **ASSOCIATIVITY** | Input | Associativity for Auxiliary Control Register |
| **CACHEID[5:0]** | Input | Cache controller cache ID |
| **CFGBIGEND** | Input | Big-endian mode for accessing configuration registers |
| **RDATALAT[2:0]** | Output | Latency for read data RAM from Auxiliary Control Register |
| **REGFILEBASE[19:0]** | Input | Base address for accessing configuration registers |
| **SE** | Input | DFT test enable, held HIGH during serial shift of scan chains and LOW for capture |
| **TAGLAT[2:0]** | Output | Latency for tag RAM from Auxiliary Control Register |
| **WAYSIZE[2:0]** | Input | Size of ways for Auxiliary Control Register |
| **WDATALAT[2:0]** | Output | Latency for write data RAM from Auxiliary Control Register |

## A.3 Slave and master ports

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

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

### A.3.1 Slave port 0

Table A-3 shows the slave port 0 signals.

**Table A-3 Slave port 0 signals**

| Signal | Type | Description |
|---|---|---|
| **ARADDRS0[31:0]** | Input | Address bus |
| **ARBURSTS0[1:0]** | Input | Burst type |
| **ARCACHES0[3:0]** | Input | Cache information |
| **ARIDS0[`pl310_AXI_ID_MAX:0]** | Input | Address ID |
| **ARLENS0[3:0]** | Input | Burst length |
| **ARLOCKS0[1:0]** | Input | Lock type |
| **ARPROTS0[2:0]** | Input | Protection information |
| **ARREADYS0** | Output | Address accepted |
| **ARSIZES0[2:0]** | Input | Burst size |
| **ARUSERS0[7:5]** | Input | Master ID for lockdown by master |
| **ARUSERS0[4:1]** | Input | Inner cacheable attributes |
| **ARUSERS0[0]** | Input | Shared attribute |
| **ARVALIDS0** | Input | Address valid |
| **AWADDRS0[31:0]** | Input | Address bus |
| **AWBURSTS0[1:0]** | Input | Burst type |
| **AWCACHES0[3:0]** | Input | Cache information |
| **AWIDS0[`pl310_AXI_ID_MAX:0]** | Input | Address ID |

**Table A-3 Slave port 0 signals (continued)**

| Signal | Type | Description |
|---|---|---|
| **AWLENS0[3:0]** | Input | Burst length |
| **AWLOCKS0[1:0]** | Input | Lock type |
| **AWPROTS0[2:0]** | Input | Protection information |
| **AWREADYS0** | Output | Address accepted |
| **AWSIZES0[2:0]** | Input | Burst size |
| **AWUSERS0[9]** | Input | Indicates a clean eviction[a] |
| **AWUSERS0[8]** | Input | Indicates an eviction[a] |
| **AWUSERS0[7:5]** | Input | Master ID for lockdown by master |
| **AWUSERS0[4:1]** | Input | Inner cacheable attributes |
| **AWUSERS0[0]** | Input | Shared attribute |
| **AWVALIDS0** | Input | Address valid |
| **BIDS0[`pl310_AXI_ID_MAX:0]** | Output | Write ID |
| **BREADYS0** | Input | Write response accepted |
| **BRESPS0[1:0]** | Output | Write response |
| **BVALIDS0** | Output | Write response valid |
| **RDATAS0[63:0]** | Output | Read data bus |
| **RIDS0[`pl310_AXI_ID_MAX:0]** | Output | Read ID |
| **RLASTS0** | Output | Read last transfer |
| **RREADYS0** | Input | Read accepted |
| **RRESPS0[1:0]** | Output | Read response |
| **RVALIDS0** | Output | Read data valid |
| **WDATAS0[63:0]** | Input | Write data bus |
| **WIDS0[`pl310_AXI_ID_MAX:0]** | Input | Write ID |
| **WLASTS0** | Input | Write last transfer |

**Table A-3 Slave port 0 signals (continued)**

| Signal | Type | Description |
|---|---|---|
| **WREADYS0** | Output | Write data accepted |
| **WSTRBS0[7:0]** | Input | Write strobes |
| **WVALIDS0** | Input | Write data valid |

a. Exclusive cache configuration only.

### A.3.2 Slave port 1

Table A-4 shows the slave port 1 signals.

**Table A-4 Slave port 1 signals**

| Signal | Type | Description |
|---|---|---|
| **ARADDRS1[31:0]** | Input | Address bus |
| **ARBURSTS1[1:0]** | Input | Burst type |
| **ARCACHES1[3:0]** | Input | Cache information |
| **ARIDS1[`pl310_AXI_ID_MAX:0]** | Input | Address ID |
| **ARLENS1[3:0]** | Input | Burst length |
| **ARLOCKS1[1:0]** | Input | Lock type |
| **ARPROTS1[2:0]** | Input | Protection information |
| **ARREADYS1** | Output | Address accepted |
| **ARSIZES1[2:0]** | Input | Burst size |
| **ARUSERS1[7:5]** | Input | Master ID for lockdown by master |
| **ARUSERS1[4:1]** | Input | Inner cacheable attributes |
| **ARUSERS1[0]** | Input | Shared attribute |
| **ARVALIDS1** | Input | Address valid |
| **AWADDRS1[31:0]** | Input | Address bus |
| **AWBURSTS1[1:0]** | Input | Burst type |
| **AWCACHES1[3:0]** | Input | Cache information |

**Table A-4 Slave port 1 signals (continued)**

| Signal | Type | Description |
|---|---|---|
| **AWIDS1[`pl310_AXI_ID_MAX:0]** | Input | Address ID |
| **AWLENS1[3:0]** | Input | Burst length |
| **AWLOCKS1[1:0]** | Input | Lock type |
| **AWPROTS1[2:0]** | Input | Protection information |
| **AWREADYS1** | Output | Address accepted |
| **AWSIZES1[2:0]** | Input | Burst size |
| **AWUSERS1[9]** | Input | Indicates a clean eviction[a] |
| **AWUSERS1[8]** | Input | Indicates an eviction[a] |
| **AWUSERS1[7:5]** | Input | Master ID for lockdown by master |
| **AWUSERS1[4:1]** | Input | Inner cacheable attributes |
| **AWUSERS1[0]** | Input | Shared attribute |
| **AWVALIDS1** | Input | Address valid |
| **BIDS1[`pl310_AXI_ID_MAX:0]** | Output | Write ID |
| **BREADYS1** | Input | Write response accepted |
| **BRESPS1[1:0]** | Output | Write response |
| **BVALIDS1** | Output | Write response valid |
| **RDATAS1[63:0]** | Output | Read data bus |
| **RIDS1[`pl310_AXI_ID_MAX:0]** | Output | Read ID |
| **RLASTS1** | Output | Read last transfer |
| **RREADYS1** | Input | Read accepted |
| **RRESPS1[1:0]** | Output | Read response |
| **RVALIDS1** | Output | Read data valid |
| **WDATAS1[63:0]** | Input | Write data bus |
| **WIDS1[`pl310_AXI_ID_MAX:0]** | Input | Write ID |
| **WLASTS1** | Input | Write last transfer |

**Table A-4 Slave port 1 signals (continued)**

| Signal | Type | Description |
|---|---|---|
| **WREADYS1** | Output | Write data accepted |
| **WSTRBS1[7:0]** | Input | Write strobes |
| **WVALIDS1** | Input | Write data valid |

a. Exclusive cache configuration only.

### A.3.3 Master port 0

Table A-5 shows the master port 0 signals.

**Table A-5 Master port 0 signals**

| Signal | Type | Description |
|---|---|---|
| **ARADDRM0[31:0]** | Output | Address bus |
| **ARBURSTM0[1:0]** | Output | Burst type |
| **ARCACHEM0[3:0]** | Output | Cache information |
| **ARIDM0[`pl310_AXI_ID_MAX+2:0]** | Output | Address ID |
| **ARLENM0[3:0]** | Output | Burst length |
| **ARLOCKM0[1:0]** | Output | Lock type |
| **ARPROTM0[2:0]** | Output | Protection information |
| **ARREADYM0** | Input | Address accepted |
| **ARSIZEM0[2:0]** | Output | Burst size |
| **ARVALIDM0** | Output | Address valid |
| **AWADDRM0[31:0]** | Output | Address bus |
| **AWBURSTM0[1:0]** | Output | Burst type |
| **AWCACHEM0[3:0]** | Output | Cache information |
| **AWIDM0[`pl310_AXI_ID_MAX+2:0]** | Output | Address ID |
| **AWLENM0[3:0]** | Output | Burst length |
| **AWLOCKM0[1:0]** | Output | Lock type |

**Table A-5 Master port 0 signals (continued)**

| Signal | Type | Description |
|---|---|---|
| **AWPROTM0[2:0]** | Output | Protection information |
| **AWREADYM0** | Input | Address accepted |
| **AWSIZEM0[2:0]** | Output | Burst size |
| **AWVALIDM0** | Output | Address valid |
| **BIDM0[`pl310_AXI_ID_MAX+2:0]** | Input | Write ID |
| **BREADYM0** | Output | Write response accepted |
| **BRESPM0[1:0]** | Input | Write response |
| **BVALIDM0** | Input | Write response valid |
| **RDATAM0[63:0]** | Input | Read data bus |
| **RIDM0[`pl310_AXI_ID_MAX+2:0]** | Input | Read ID |
| **RLASTM0** | Input | Read last transfer |
| **RREADYM0** | Output | Read accepted |
| **RRESPM0[1:0]** | Input | Read response |
| **RVALIDM0** | Input | Read data valid |
| **WDATAM0[63:0]** | Output | Write data bus |
| **WIDM0[`pl310_AXI_ID_MAX+2:0]** | Output | Write ID |
| **WLASTM0** | Output | Write last transfer |
| **WREADYM0** | Input | Write data accepted |
| **WSTRBM0[7:0]** | Output | Write strobes |
| **WVALIDM0** | Output | Write data valid |

### A.3.4 Master port 1

Table A-6 shows the master port 1 signals.

**Table A-6 Master port 1 signals**

| Signal | Type | Description |
|---|---|---|
| **ARADDRM1[31:0]** | Output | Address bus |
| **ARBURSTM1[1:0]** | Output | Burst type |
| **ARCACHEM1[3:0]** | Output | Cache information |
| **ARIDM1[`pl310_AXI_ID_MAX+2:0]** | Output | Address ID |
| **ARLENM1[3:0]** | Output | Burst length |
| **ARLOCKM1[1:0]** | Output | Lock type |
| **ARPROTM1[2:0]** | Output | Protection information |
| **ARREADYM1** | Input | Address accepted |
| **ARSIZEM1[2:0]** | Output | Burst size |
| **ARVALIDM1** | Output | Address valid |
| **AWADDRM1[31:0]** | Output | Address bus |
| **AWBURSTM1[1:0]** | Output | Burst type |
| **AWCACHEM1[3:0]** | Output | Cache information |
| **AWIDM1[`pl310_AXI_ID_MAX+2:0]** | Output | Address ID |
| **AWLENM1[3:0]** | Output | Burst length |
| **AWLOCKM1[1:0]** | Output | Lock type |
| **AWPROTM1[2:0]** | Output | Protection information |
| **AWREADYM1** | Input | Address accepted |
| **AWSIZEM1[2:0]** | Output | Burst size |
| **AWVALIDM1** | Output | Address valid |
| **BIDM1[`pl310_AXI_ID_MAX+2:0]** | Input | Write ID |
| **BREADYM1** | Output | Write response accepted |
| **BRESPM1[1:0]** | Input | Write response |

**Table A-6 Master port 1 signals (continued)**

| Signal | Type | Description |
|---|---|---|
| **BVALIDM1** | Input | Write response valid |
| **RDATAM1[63:0]** | Input | Read data bus |
| **RIDM1[`pl310_AXI_ID_MAX+2:0]** | Input | Read ID |
| **RLASTM1** | Input | Read last transfer |
| **RREADYM1** | Output | Read accepted |
| **RRESPM1[1:0]** | Input | Read response |
| **RVALIDM1** | Input | Read data valid |
| **WDATAM1[63:0]** | Output | Write data bus |
| **WIDM1[`pl310_AXI_ID_MAX+2:0]** | Output | Write ID |
| **WLASTM1** | Output | Write last transfer |
| **WREADYM1** | Input | Write data accepted |
| **WSTRBM1[7:0]** | Output | Write strobes |
| **WVALIDM1** | Output | Write data valid |

## A.4 RAM interface

The RAM interface consists of three sub interfaces:

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

### A.4.1 Data RAM interface

Table A-7 shows the data RAM interface signals.

**Table A-7 Data RAM interface signals**

| Signal | Type | Description |
|---|---|---|
| **DATAADDR[17:0]**[a]<br>**DATAADDR[16:0]**[b] | Output | Data RAM address |
| **DATACS** | Output | Data RAM chip select |
| **DATAEN[31:0]** | Output | Data RAM byte write enables |
| **DATAERR** | Input | Data RAM error |
| **DATAnRW** | Output | Data RAM write control signal |
| **DATAPRD[31:0]**[c] | Input | Data RAM parity read data |
| **DATAPWD[31:0]**[c] | Output | Data RAM parity write data |
| **DATARD[255:0]** | Input | Data RAM read data |
| **DATAWAIT** | Input | Data RAM wait |
| **DATAWD[255:0]** | Output | Data RAM write data |

a. For 16-way implementation.
b. For 8-way implementation.
c. Optional. Only present if `pl310_PARITY` is defined.

### A.4.2 Tag RAM interface

Table A-8 shows the tag RAM interface signals.

**Table A-8 Tag RAM interface**

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

**Table A-8 Tag RAM interface (continued)**

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

a. For 16-way implementation.
b. For 8-way implementation.
c. Optional. Only present if `pl310_LOCKDOWN_BY_LINE` is defined.
d. Optional. Only present if `pl310_PARITY` is defined.
e. Only present for 16-way implementation.

## A.5 Cache event monitoring

Table A-9 shows the cache event monitoring signals.

**Table A-9 Cache event monitoring signals**

| Signal | Type | Description |
|---|---|---|
| **CO** | Output | Eviction (CastOut) of a line from the L2 cache. |
| **DRHIT** | Output | Data read hit |
| **DRREQ** | Output | Data read request |
| **DWHIT** | Output | Data write hit |
| **DWREQ** | Output | Data write request |
| **DWTREQ** | Output | Data write request with write-through attribute |
| **IRHIT** | Output | Instruction read hit |
| **IRREQ** | Output | Instruction read request |
| **SPNIDEN** | Input | Secure privileged non-invasive debug enable. |
| **WA** | Output | Write allocate (write transaction cause a linefill) |

## A.6 Cache interrupt

Table A-10 shows the cache interrupt signals.

**Table A-10 Cache Interrupt signals**

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

## A.7 MBIST interface

Table A-11 shows the MBIST interface signals.

**Table A-11 MBIST interface signals**

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

# Appendix B
# **AC Parameters**

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



**Note**

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

## B.1 Reset and configuration signal timing parameters

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

**Table B-1 Reset and configuration**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **nRESETn** | Input | 20% |
| **CACHEID[5:0]** | Input | 20% |
| **TAGLAT[2:0]** | Output | 70% |
| **WDATALAT[2:0]** | Output | 70% |
| **RDATALAT[2:0]** | Output | 70% |
| **SCANENABLE** | Input | 20% |
| **ASSOCIATIVITY[3:0]** | Input | 20% |
| **WAYSIZE[2:0]** | Input | 20% |
| **CFGBIGEND** | 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 |
|---|---|---|
| **ARADDRS0[31:0]** | Input | 50% |
| **ARBURSTS0[1:0]** | Input | 70% |
| **ARCACHES0[3:0]** | Input | 50% |
| **ARIDS0[5:0]** | Input | 50% |
| **ARLENS0[3:0]** | Input | 70% |
| **ARLOCKS0[1:0]** | Input | 50% |
| **ARPROTS0[2:0]** | Input | 70% |
| **ARREADYS0** | Output | 70% |
| **ARSIZES0[2:0]** | Input | 70% |
| **ARUSERS0[4:0]** | Input | 50% |
| **ARVALIDS0** | Input | 50% |
| **AWADDRS0[31:0]** | Input | 50% |
| **AWBURSTS0[1:0]** | Input | 70% |
| **AWCACHES0[3:0]** | Input | 70% |
| **AWIDS0[5:0]** | Input | 70% |
| **AWLENS0[3:0]** | Input | 70% |
| **AWLOCKS0[1:0]** | Input | 50% |
| **AWPROTS0[2:0]** | Input | 70% |
| **AWREADYS0** | Output | 70% |
| **AWSIZES0[2:0]** | Input | 70% |
| **AWUSERS0[4:0]** | Input | 70% |
| **AWUSERS0[9:8]** | Input | 70% |
| **AWVALIDS0** | Input | 50% |

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

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **BIDS0[5:0]** | Output | 70% |
| **BREADYS0** | Input | 70% |
| **BRESPS0[1:0]** | Output | 70% |
| **BVALIDS0** | Output | 70% |
| **INCLKENS0** | Input | 30% |
| **OUTCLKENS0** | Output | 30% |
| **RDATAS0[63:0]** | Output | 70% |
| **RIDS0[5:0]** | Output | 70% |
| **RLASTS0** | Output | 70% |
| **RREADYS0** | Input | 50% |
| **RRESPS0[1:0]** | Output | 50% |
| **RVALIDS0** | Output | 70% |
| **WDATAS0[63:0]** | Input | 70% |
| **WIDS0[5:0]** | Input | 70% |
| **WLASTS0** | Input | 50% |
| **WREADYS0** | Output | 70% |
| **WSTRBS0[7:0]** | Input | 50% |
| **WVALIDS0** | Input | 50% |

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

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

**Table B-3 Slave port 1 I/O**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **ARADDRS1[31:0]** | Input | 50% |
| **ARBURSTS1[1:0]** | Input | 70% |
| **ARCACHES1[3:0]** | Input | 50% |
| **ARIDS1[5:0]** | Input | 50% |
| **ARLENS1[3:0]** | Input | 70% |
| **ARLOCKS1[1:0]** | Input | 50% |
| **ARPROTS1[2:0]** | Input | 70% |
| **ARREADYS1** | Output | 70% |
| **ARSIZES1[2:0]** | Input | 70% |
| **ARUSERS1[4:0]** | Input | 50% |
| **ARVALIDS1** | Input | 50% |
| **AWADDRS1[31:0]** | Input | 50% |
| **AWBURSTS1[1:0]** | Input | 70% |
| **AWCACHES1[3:0]** | Input | 70% |
| **AWIDS1[5:0]** | Input | 70% |
| **AWLENS1[3:0]** | Input | 70% |
| **AWLOCKS1[1:0]** | Input | 50% |
| **AWPROTS1[2:0]** | Input | 70% |
| **AWREADYS1** | Output | 70% |
| **AWSIZES1[2:0]** | Input | 70% |
| **AWUSERS1[4:0]** | Input | 70% |
| **AWUSERS1[9:8]** | Input | 70% |
| **AWVALIDS1** | Input | 50% |

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

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **BIDS1[5:0]** | Output | 70% |
| **BREADYS1** | Input | 70% |
| **BRESPS1[1:0]** | Output | 70% |
| **BVALIDS1** | Output | 70% |
| **INCLKENS1** | Input | 30% |
| **OUTCLKENS1** | Output | 30% |
| **RDATAS1[63:0]** | Output | 70% |
| **RIDS1[5:0]** | Output | 70% |
| **RLASTS1** | Output | 70% |
| **RREADYS1** | Input | 50% |
| **RRESPS1[1:0]** | Output | 50% |
| **RVALIDS1** | Output | 70% |
| **WDATAS1[63:0]** | Input | 70% |
| **WIDS1[5:0]** | Input | 70% |
| **WLASTS1** | Input | 50% |
| **WREADYS1** | Output | 70% |
| **WSTRBS1[7:0]** | Input | 50% |
| **WVALIDS1** | Input | 50% |

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

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

**Table B-4 Master port 0 I/O**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **ARADDRM0[31:0]** | Output | 70% |
| **ARBURSTM0[1:0]** | Output | 70% |
| **ARCACHEM0[3:0]** | Output | 70% |
| **ARESETM0n** | Input | 20% |
| **ARIDM0[7:0]** | Output | 70% |
| **ARLENM0[3:0]** | Output | 70% |
| **ARLOCKM0[1:0]** | Output | 70% |
| **ARPROTM0[2:0]** | Output | 70% |
| **ARREADYM0** | Input | 70% |
| **ARSIZEM0[2:0]** | Output | 70% |
| **ARUSERM0[4:0]** | Output | 50% |
| **ARVALIDM0** | Output | 70% |
| **AWADDRM0[31:0]** | Output | 70% |
| **AWBURSTM0[1:0]** | Output | 70% |
| **AWCACHEM0[3:0]** | Output | 70% |
| **AWIDM0[7:0]** | Output | 70% |
| **AWLENM0[3:0]** | Output | 70% |
| **AWLOCKM0[1:0]** | Output | 70% |
| **AWPROTM0[2:0]** | Output | 70% |
| **AWREADYM0** | Input | 50% |
| **AWSIZEM0[2:0]** | Output | 70% |
| **AWUSERM0[4:0]** | Output | 70% |
| **AWVALIDM0** | Output | 700% |

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

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **BIDM0[7:0]** | Input | 50% |
| **BREADYM0** | Output | 70% |
| **BRESPM0[1:0]** | Input | 50% |
| **BVALIDM0** | Input | 50% |
| **INCLKENM0** | Input | 30% |
| **OUTCLKENM0** | Output | 30% |
| **RDATAM0[63:0]** | Input | 70% |
| **RIDM0[7:0]** | Input | 50% |
| **RLASTM0** | Input | 50% |
| **RREADYM0** | Output | 70% |
| **RRESPM0[1:0]** | Input | 50% |
| **RVALIDM0** | Input | 50% |
| **WDATAM0[63:0]** | Output | 70% |
| **WIDM0[7:0]** | Output | 70% |
| **WLASTM0** | Output | 70% |
| **WREADYM0** | Input | 50% |
| **WSTRBM0[7:0]** | Output | 70% |
| **WVALIDM0** | Output | 50% |

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

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

**Table B-5 Master port 1 I/O**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **ARADDRM1[31:0]** | Output | 70% |
| **ARBURSTM1[1:0]** | Output | 70% |
| **ARCACHEM1[3:0]** | Output | 70% |
| **ARESETM1n** | Input | 20% |
| **ARIDM1[7:0]** | Output | 70% |
| **ARLENM1[3:0]** | Output | 70% |
| **ARLOCKM1[1:0]** | Output | 70% |
| **ARPROTM1[2:0]** | Output | 70% |
| **ARREADYM1** | Input | 70% |
| **ARSIZEM1[2:0]** | Output | 70% |
| **ARUSERM1[4:0]** | Output | 50% |
| **ARVALIDM1** | Output | 70% |
| **AWADDRM1[31:0]** | Output | 70% |
| **AWBURSTM1[1:0]** | Output | 70% |
| **AWCACHEM1[3:0]** | Output | 70% |
| **AWIDM1[7:0]** | Output | 70% |
| **AWLENM1[3:0]** | Output | 70% |
| **AWLOCKM1[1:0]** | Output | 70% |
| **AWPROTM1[2:0]** | Output | 70% |
| **AWREADYM1** | Input | 50% |
| **AWSIZEM1[2:0]** | Output | 70% |
| **AWUSERM1[4:0]** | Output | 70% |
| **AWVALIDM1** | Output | 700% |

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

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **BIDM1[7:0]** | Input | 50% |
| **BREADYM1** | Output | 70% |
| **BRESPM1[1:0]** | Input | 50% |
| **BVALIDM1** | Input | 50% |
| **INCLKENM1** | Input | 30% |
| **OUTCLKENM1** | Output | 30% |
| **RDATAM1[63:0]** | Input | 70% |
| **RIDM1[7:0]** | Input | 50% |
| **RLASTM1** | Input | 50% |
| **RREADYM1** | Output | 70% |
| **RRESPM1[1:0]** | Input | 50% |
| **RVALIDM1** | Input | 50% |
| **WDATAM1[63:0]** | Output | 70% |
| **WIDM1[7:0]** | Output | 70% |
| **WLASTM1** | Output | 70% |
| **WREADYM1** | Input | 50% |
| **WSTRBM1[7:0]** | Output | 70% |
| **WVALIDM1** | Output | 50% |

## B.6 RAMs signal timing parameters

This section shows the RAMs signal timing parameters in:

- *Data RAM*
- *Tag RAM*.

### B.6.1 Data RAM

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

**Table B-6 Data RAM**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **DATAADDR[16:0]** | Output | 70% |
| **DATACS** | Output | 70% |
| **DATAEN[31:0]** | Output | 70% |
| **DATAERR** | Input | 50% |
| **DATAnRW** | Output | 70% |
| **DATAPEN[31:0]** | Output | 70% |
| **DATAPnRW** | Output | 70% |
| **DATAPRD[31:0]** | Input | 50% |
| **DATAPWD[31:0]** | Output | 50% |
| **DATARD[255:0]** | Input | 50% |
| **DATAWD[255:0]** | Output | 50% |

### B.6.2 Tag RAM

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

**Table B-7 Tag RAM**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **TAGADDR[12:0]** | Output | 70% |
| **TAGCS[15:0]** | Output | 70% |
| **TAGEN[20:0]** | Output | 70% |

**Table B-7 Tag RAM (continued)**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **TAGLEN** | Output | 70% |
| **TAGERR[7:0]** | Input | 70% |
| **TAGPRD[7:0]** | Input | 70% |
| **TAGPWD** | Output | 60% |
| **TAGRD0[20:0]** | Input | 70% |
| **TAGLRD[15:0]** | Input | 70% |
| **TAGLWD** | Output | 70% |
| **TAGRD1[20:0]** | Input | 70% |
| **TAGRD2[20:0]** | Input | 70% |
| **TAGRD3[20:0]** | Input | 70% |
| **TAGRD4[20:0]** | Input | 70% |
| **TAGRD5[20:0]** | Input | 70% |
| **TAGRD6[20:0]** | Input | 70% |
| **TAGRD7[20:0]** | Input | 70% |
| **TAGRD8[20:0]** | Input | 70% |
| **TAGRD9[20:0]** | Input | 70% |
| **TAGRD10[20:0]** | Input | 70% |
| **TAGRD11[20:0]** | Input | 70% |
| **TAGRD12[20:0]** | Input | 70% |
| **TAGRD13[20:0]** | Input | 70% |
| **TAGRD14[20:0]** | Input | 70% |
| **TAGRD15[20:0]** | Input | 70% |
| **TAGWD[19:0]** | Output | 60% |

## B.7 Event monitor signal timing parameters

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

**Table B-8 Event monitor**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **CO** | Output | 70% |
| **DRHIT** | Output | 70% |
| **DRREQ** | Output | 70% |
| **DWHIT** | Output | 70% |
| **DWREQ** | Output | 70% |
| **DWTREQ** | Output | 70% |
| **IRHIT** | Output | 70% |
| **IRREQ** | Output | 70% |
| **SPNIDEN** | Input | 70% |
| **WA** | Output | 70% |

## B.8 Cache interrupt ports signal timing parameters

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

**Table B-9 Cache interrupt ports**

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

## B.9 MBIST interface signal timing parameters

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

**Table B-10 MBIST interface signal**

| Port name | I/O type | Maximum constraint |
|---|---|---|
| **MBISTADDR[18:0]** | Input | 70% |
| **MBISTCE[17:0]** | Input | 70% |
| **MBISTDCTL[19:0]** | Output | 70% |
| **MBISTDIN[63:0]** | Input | 70% |
| **MBISTWE** | Input | 70% |
| **MBISTON** | Input | 30% |
| **MBISTDOUT[63:0]** | Output | 50% |

# Appendix C
# **Timing Diagrams**

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



**Note**

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

## C.1 Single read hit transaction

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

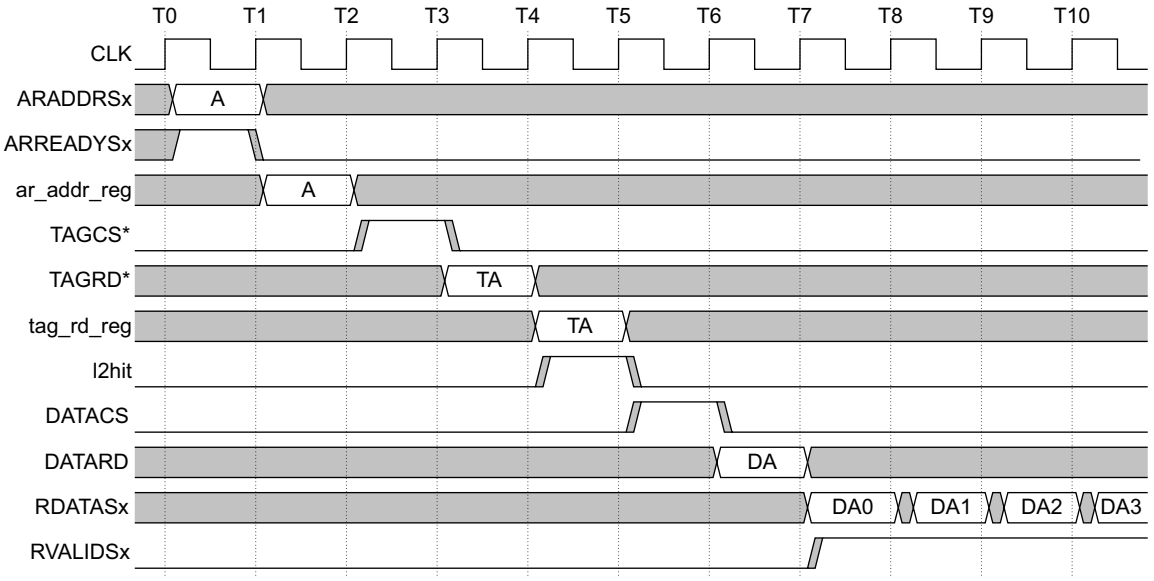



**Figure C-1 Single read hit transaction**

## C.2 Single read miss transaction

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

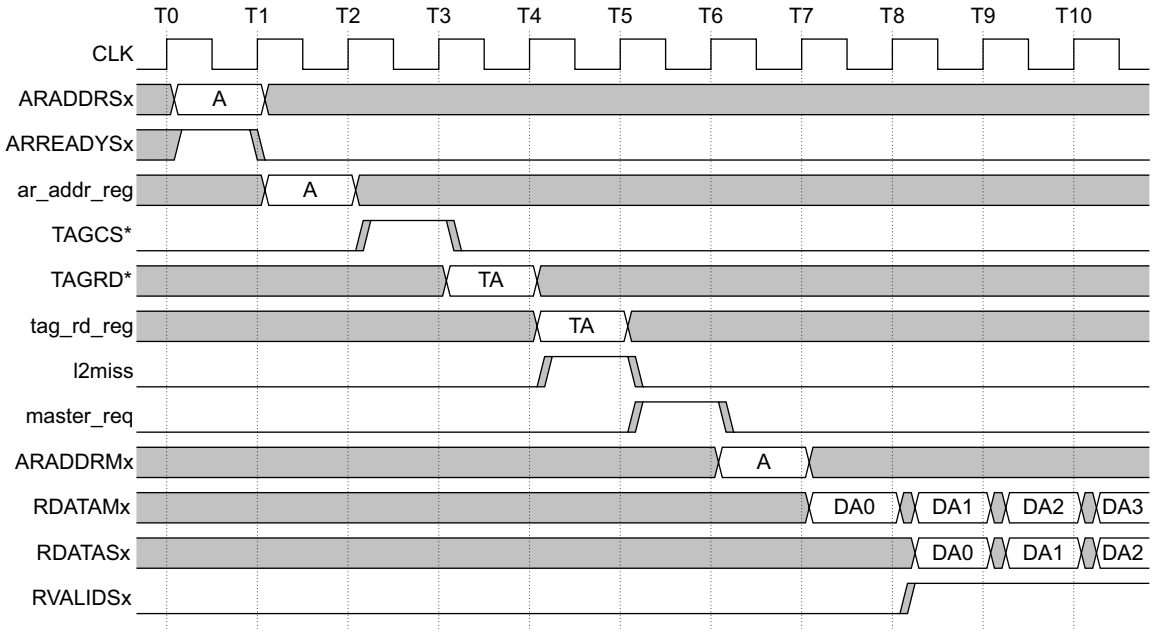



**Figure C-2 Single read miss transaction**

## C.3 Single noncacheable read transaction

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

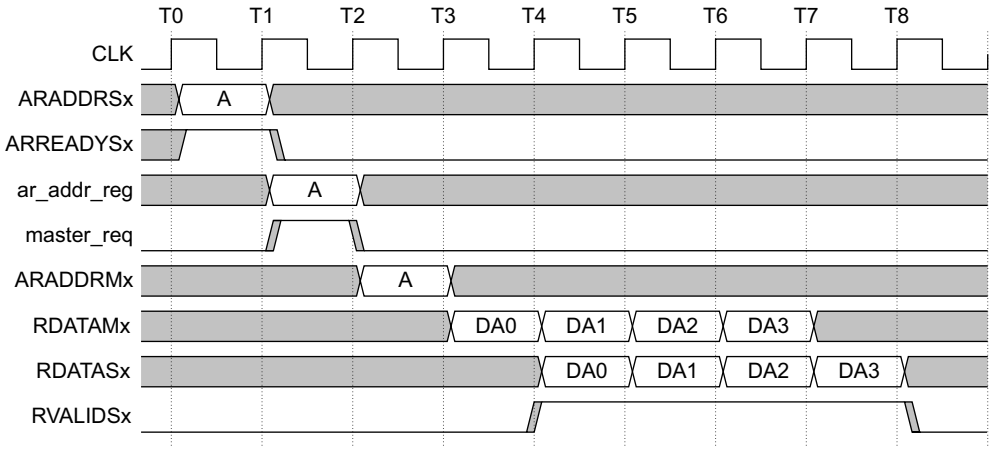



**Figure C-3 Single noncacheable read transaction**

## C.4 Outstanding read hit transaction

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

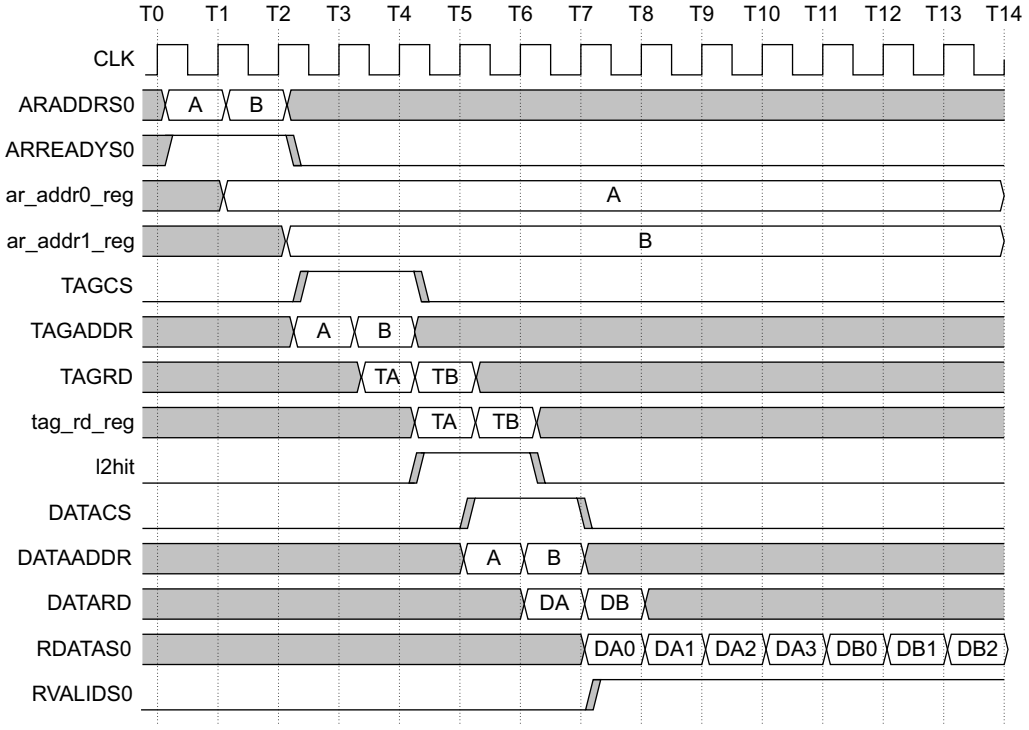



**Figure C-4 Outstanding read hit transaction**

## C.5 Hit under miss read transactions

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

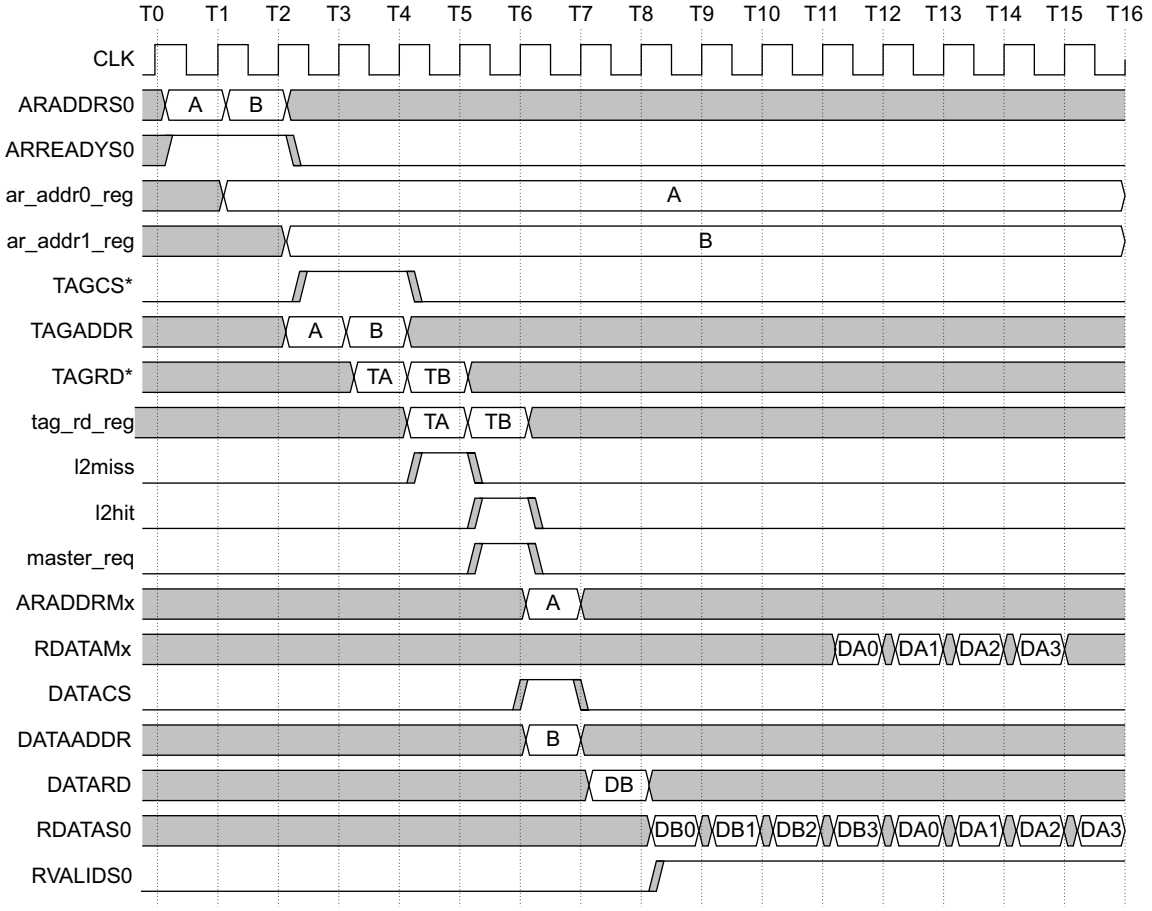



**Figure C-5 Hit under miss read transaction**

## C.6 Single bufferable write transaction

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

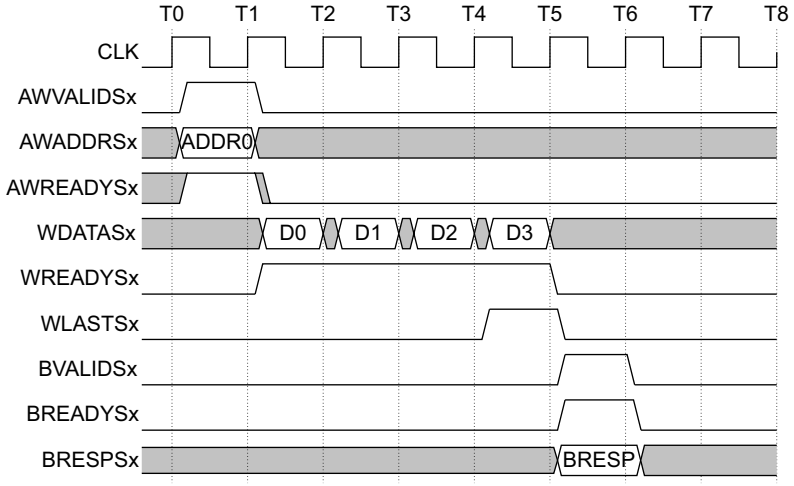



**Figure C-6 Single bufferable write transaction**

## C.7 Single nonbufferable write transaction

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

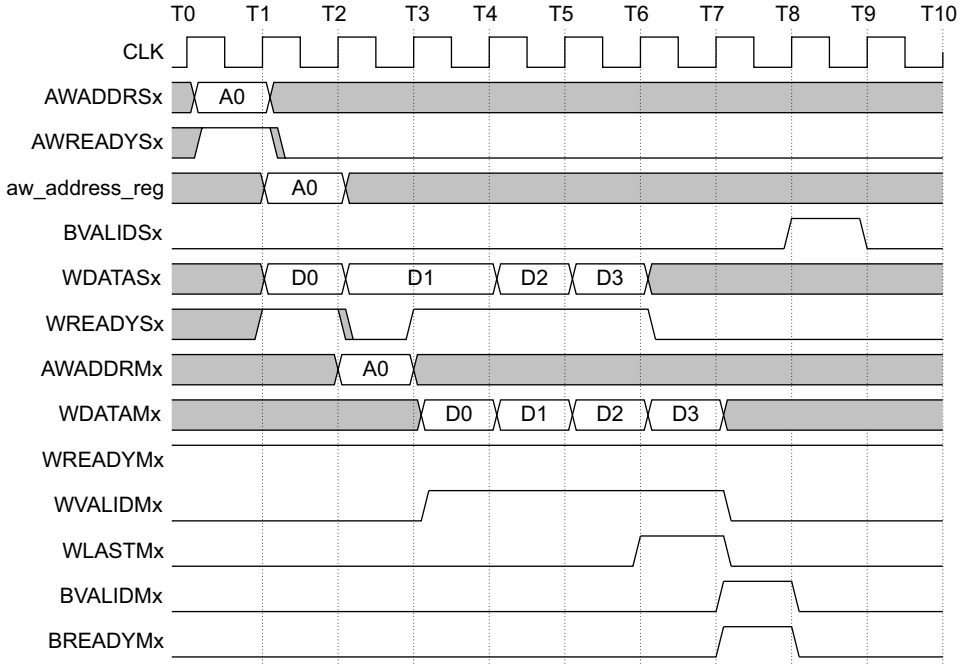



**Figure C-7 Single nonbufferable write transaction**

# Glossary

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

**Abort**

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

*See also* Data Abort.

**Advanced eXtensible Interface (AXI)**

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

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

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

**AMBA**
: *See* Advanced Microcontroller Bus Architecture.

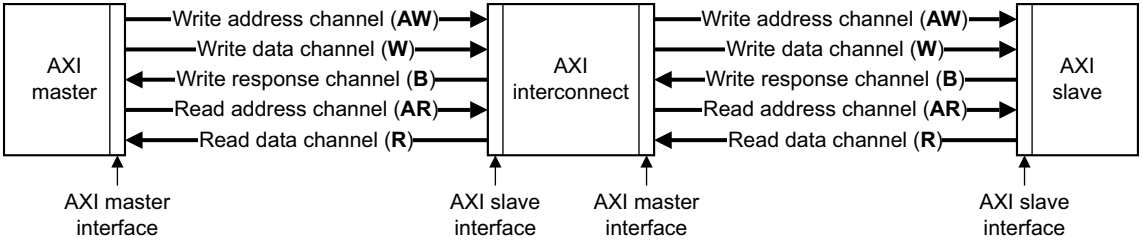
**Application Specific Integrated Circuit (ASIC)**
: An integrated circuit that has been designed to perform a specific application function. It can be custom-built or mass-produced.

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

**ASIC**
: *See* Application Specific Integrated Circuit.

**AXI**
: *See* Advanced eXtensible Interface.

**AXI channel order and interfaces**
: The block diagram shows:
  - the order in which AXI channel signals are described
  - the master and slave interface conventions for AXI components.




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

  **Active read transaction**

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

  **Active transfer**

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

**Active write transaction**

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

**Completed transfer**

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

**Payload** The non-handshake signals in a transfer.

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

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

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

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

**Combined issuing capability**

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

**Read ID capability**

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

**Read ID width**

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

**Read issuing capability**

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

---

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

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

**Write ID capability**

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

**Write ID width**

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

**Write interleave capability**

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

**Write issuing capability**

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

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

**Combined acceptance capability**

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

**Read acceptance capability**

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

**Read data reordering depth**

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

**Write acceptance capability**

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

**Write interleave depth**

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

**Beat**

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

*See also* Burst.

**Big-endian** 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.

**Coherency** *See* Memory coherency.

**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.

**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.

**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.

**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.

**Memory coherency** A memory is coherent if the value read by a data read or instruction fetch is the value that was most recently written to that location. Memory coherency is made difficult when there are multiple possible physical locations that are involved, such as a system that has main memory, a store buffer, and a cache.

**Microprocessor** *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.

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

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

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

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

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

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

  *See also* Aligned.

**UNP**
: *See* Unpredictable.

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

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

**Way**
: *See* Cache way.

**WB**
: *See* Write-back.

**Word**
: A 32-bit data item.

**Write** Writes are defined as operations that have the semantics of a store. That is, the ARM instructions SRS, STM, STRD, STC, STRT, STRH, STRB, STRBT, STREX, SWP, and SWPB, and the Thumb instructions STM, STR, STRH, STRB, and PUSH. Java instructions that are accelerated by hardware can cause a number of writes to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.

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

**Write completion** 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.