CoreLink™ CCI-400 Cache Coherent Interconnect

Revision: r0p3

Technical Reference Manual

ARM®

ARM DDI 0470D (ID040512)

CoreLink CCI-400 Cache Coherent Interconnect
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
16 May 2011	A	Non-Confidential	First release for r0p0
07 July 2011	B	Non-Confidential	First release for r0p1
16 September 2011	C	Non-Confidential	First release for r0p2
12 March 2012	D	Non-Confidential	First release for r0p3

Proprietary Notice

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Contents
CoreLink CCI-400 Cache Coherent Interconnect Technical Reference Manual

Preface

This preface introduces the *CoreLink™ CCI-400 Cache Coherent Interconnect Technical Reference Manual* (TRM). It contains the following sections:

About this book

This book is for CoreLink™ CCI-400 Cache Coherent Interconnect.

Product revision status

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

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

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

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the CoreLink CCI-400 Cache Coherent Interconnect.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level view of the CoreLink CCI-400 Cache Coherent Interconnect and a description of its features.

Chapter 2 *Functional Description*

Read this for a description of the major interfaces and components of the CoreLink CCI-400 Cache Coherent Interconnect. This chapter also describes how the components operate.

Chapter 3 *Programmers Model*

Read this for a description of the address map and registers of the CoreLink CCI-400 Cache Coherent Interconnect.

Appendix A *Signal Descriptions*

Read this for a description of the signals that the CoreLink CCI-400 Cache Coherent Interconnect uses.

Appendix B *Revisions*

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

Glossary

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

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

Conventions

Conventions that this book can use are described in:

- *Typographical* on page vii.
- *Signals* on page vii.

Typographical

The typographical conventions are:

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

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means: • HIGH for active-HIGH signals • LOW for active-LOW signals.
Lower-case n	At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

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

ARM publications

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

- *CoreLink™ CCI-400 Cache Coherent Interconnect Integration Manual* (ARM DII 0264)
- *CoreLink CCI-400 Cache Coherent Interconnect Implementation Guide* (ARM DII 0263)
- *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022)
- *ARMv7-M Architecture Reference Manual* (ARM DDI 0403)
- *ARM Architecture Reference Manual ARMv7-A and ARMv7-R Edition* (ARM DDI 0406).

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

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

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

Feedback on content

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

- the title
- the number, ARM DDI 0470D
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the CoreLink™ CCI-400 Cache Coherent Interconnect. It contains the following sections:

1.1 About the CoreLink Cache Coherent Interconnect

The CCI-400 combines interconnect and coherency functions into a single module. It supports connectivity for up to two ACE masters, for example, Cortex-A15, and three ACE-Lite masters, for example, Mali-T604, and optional *Distributed Virtual Memory* (DVM) message support on these interfaces to manage distributed *Memory Management Units* (MMUs), for example, MMU-400. These can communicate through the CCI-400 with up to three ACE-Lite slaves.

Hardware-managed coherency can improve system performance and reduce system power by sharing on-chip data. Managing coherency has benefits in:

- reducing external memory accesses
- reducing the software overhead.

1.2 Compliance

The CCI-400 Cache Coherent Interconnect complies with, or implements, the specifications described in:

- *ARM architecture*
- *Advanced Microcontroller Bus Architecture*.

This TRM complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

1.2.1 ARM architecture

The CCI-400 Cache Coherent Interconnect implements the ARM v7 architecture profile. See *ARMv7-M Architecture Reference Manual* and *ARM Architecture Reference Manual ARMv7-A and ARMv7-R Edition*.

1.2.2 Advanced Microcontroller Bus Architecture

This CCI-400 Cache Coherent Interconnect complies with the AMBA 4 ACE protocol. See *Additional reading* on page vii.

1.3 Features

The CCI-400 Cache Coherent Interconnect is an infrastructure component that supports:

- data coherency between up to two ACE masters, and three ACE-Lite masters with three independent *Points-of-Serialization* (PoS) and full barrier support
- high-bandwidth, cross-bar interconnect functionality between the masters and up to three slaves
- DVM message transport between masters
- *Quality-of-Service* (QoS) regulation for shaping traffic profiles
- a *Performance Monitoring Unit* (PMU) to count performance-related events
- a *Programmers View* (PV) to control the coherency and interconnect functionality.

1.4 Interfaces

Figure 1-1 shows an example top-level system using a CCI-400 Cache Coherent Interconnect.

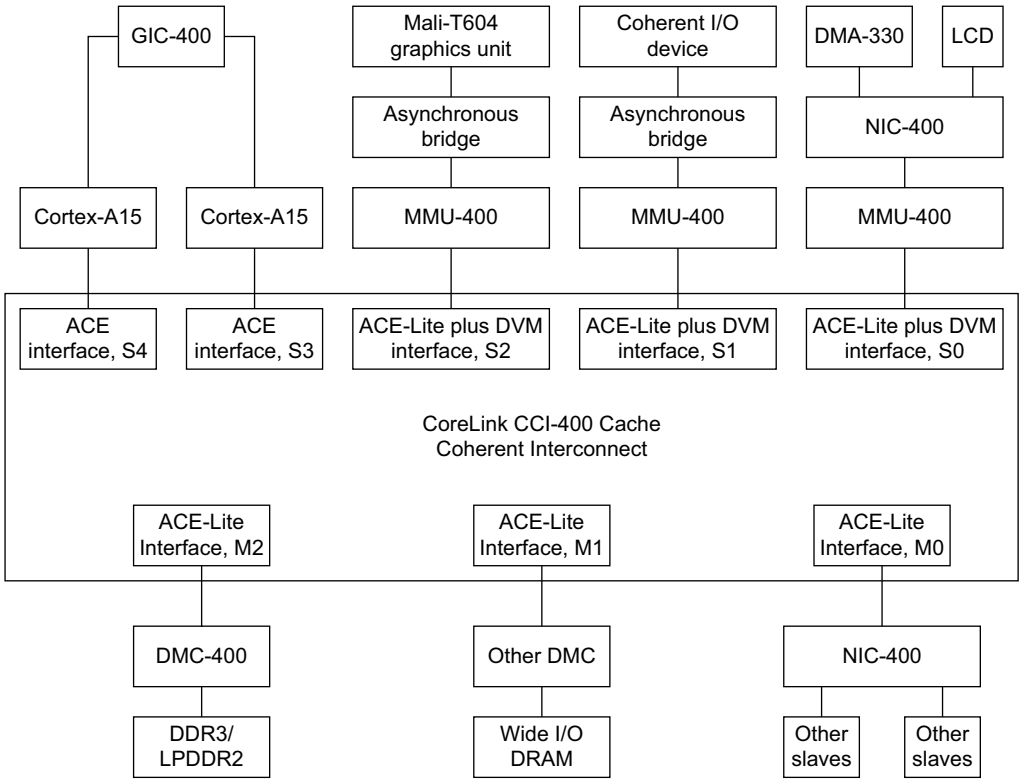

Figure 1-1 Example top-level system using a CCI-400 Cache Coherent Interconnect

Slave interfaces S4 and S3 support the ACE protocol for connection to masters such as Cortex-A15 processors. Full coherency and sharing of data is managed between these processors.

Slave interfaces S0, S1, S2 support ACE-Lite and DVM signaling for connection to I/O coherent devices such as the Mali-T604 graphics unit. You can use DVM signaling for attached MMUs such as the MMU-400.

You can use the NIC-400 Network Interconnect to connect other masters and peripherals in the system.

The ACE-Lite master interfaces M2 and M1 are optimized for connection to a compatible memory controller such as the DMC-400 Dynamic Memory Controller for DDR2, DDR3, and LPDDR2 memory.

ACE-Lite master interface M0 is designed to connect to the rest of the system.

1.5 Configurable options

The CCI-400 has design-time options for the following:

- width of ID signals on the slave interfaces
- maximum number of outstanding transactions at each slave and master interface
- registering options at each slave and master interface
- user-defined additional signaling.

The CCI-400 has reset-time options for the following:

- address map
- base address of internal registers
- master interface behavior regarding barriers and cache maintenance operations
- snoop request and DVM message propagation
- *Quality of Service* (QoS) provision.

Note

You cannot modify other parameters in the design.

Inputs exist that are sampled at reset, and that define the behavior of the CCI-400, for example, the address map. See *Address map* on page 3-20 for more information. See the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* for more information on configuring the CCI-400.

1.6 Test features

The CCI-400 Cache Coherent Interconnect contains a **DFTRSTDISABLE** input signal that disables the reset during scan shift.

1.7 Product documentation, design flow, and architecture

This section describes the CCI-400 Cache Coherent Interconnect books and how they relate to the design flow. It includes:

- *Documentation*
- *Design flow* on page 1-9.

See *Additional reading* on page vii for more information about the books described in this section. For information on the relevant architectural standards and protocols, see *Compliance* on page 1-3.

1.7.1 Documentation

The CCI-400 Cache Coherent Interconnect documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the CCI-400 Cache Coherent Interconnect. It is required at all stages of the design flow. The choices made in the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the CCI-400 Cache Coherent Interconnect then contact:

- the implementer to determine:
  - the build configuration of the implementation
  - what integration, if any, was performed before implementing the CCI-400 Cache Coherent Interconnect
- the integrator to determine the pin configuration of the device that you are using.

Implementation Guide

The *Implementation Guide* (IG) describes:

- the available build configuration options and related issues in selecting them
- how to configure the *Register Transfer Level* (RTL) code with the build configuration options
- the processes to sign off the configured design.

The ARM product deliverables include reference scripts and information about using them to implement your design.

The IG is a confidential book that is only available to licensees.

Integration Manual

The *Integration Manual* (IM) describes how to integrate the CCI-400 Cache Coherent Interconnect into a SoC. It includes describing the pins that the integrator must tie off to configure the macrocell for the required integration. Some of the integration is affected by the configuration options used when implementing the CCI-400 Cache Coherent Interconnect.

The IM is a confidential book that is only available to licensees.

1.7.2 Design flow

The CCI-400 Cache Coherent Interconnect is delivered as synthesizable RTL. Before you can use it in a product, it must go through the following processes:

Implementation The implementer configures and synthesizes the RTL to produce a hard macrocell.

Integration The integrator connects the implemented design into a SoC. This includes connecting it to a memory system and peripherals.

Programming This is the last process. The system programmer develops the software required to configure and initialize the CCI-400, and tests the required application software.

Each process:

- can be performed by a different party
- can include implementation and integration choices that affect the behavior and features of the CCI-400 Cache Coherent Interconnect.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the area, maximum frequency, and features of the resulting macrocell. For example, the numbers of outstanding transactions that each master and slave interface support.

Configuration inputs

The integrator configures some features of the CCI-400 Cache Coherent Interconnect by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software. For example, the **ACCHANNELEN** inputs that prevent AC requests from being emitted from an unconnected slave interface.

Software configuration

The programmer configures the CCI-400 Cache Coherent Interconnect by programming particular values into registers. This affects the behavior of the CCI-400, for example, by enabling or disabling speculative fetches.

Note

This manual refers to implementation-defined features that are applicable to build configuration options. Reference to a feature that is included means that the appropriate build and pin configuration options are selected. Reference to an enabled feature means one that has also been configured by software.

1.8 Product revisions

This section describes differences in functionality between product revisions:

r0p0 First release.

r0p0-r0p1 New exclusive access functionality. See *Exclusive accesses* on page 2-15 and the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* for more information.

r0p1-r0p2 Configuration improvements for the following:

- maximum ID width on slave interfaces increased from 8 to 28
- maximum size of read and write trackers in master interfaces increased from 32 to 128
- read and write trackers in master interfaces is now configurable in steps of 1 rather than 2^n.

Improvements for the following:

- master interface read tracker slot re-use time to reduce stalls
- slave interface PMU events redefined to improve the counting of device and non-cacheable transactions.

r0p2-r0p3 The CCI-400 now supports up to four exclusive access threads from each ACE master and has separate monitors for secure and non-secure transactions. See *Exclusive accesses* on page 2-15 and the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* for more information.

Chapter 2
Functional Description

This chapter describes the functionality of the CoreLink™ CCI-400 Cache Coherent Interconnect. It contains the following sections:

2.1 About the functions

The CCI-400 combines interconnect and coherency functions into a single module. It supports connectivity for up to two ACE masters, for example, Cortex-A15, and three ACE-Lite masters, for example, Mali-T604, plus optional *Distributed Virtual Memory* (DVM) message support on these interfaces to manage distributed *Memory Management Units* (MMUs), for example, MMU-400. These can communicate through the CCI-400 with up to three ACE-Lite slaves.

Figure 2-1 shows the internal architecture of the CCI-400.

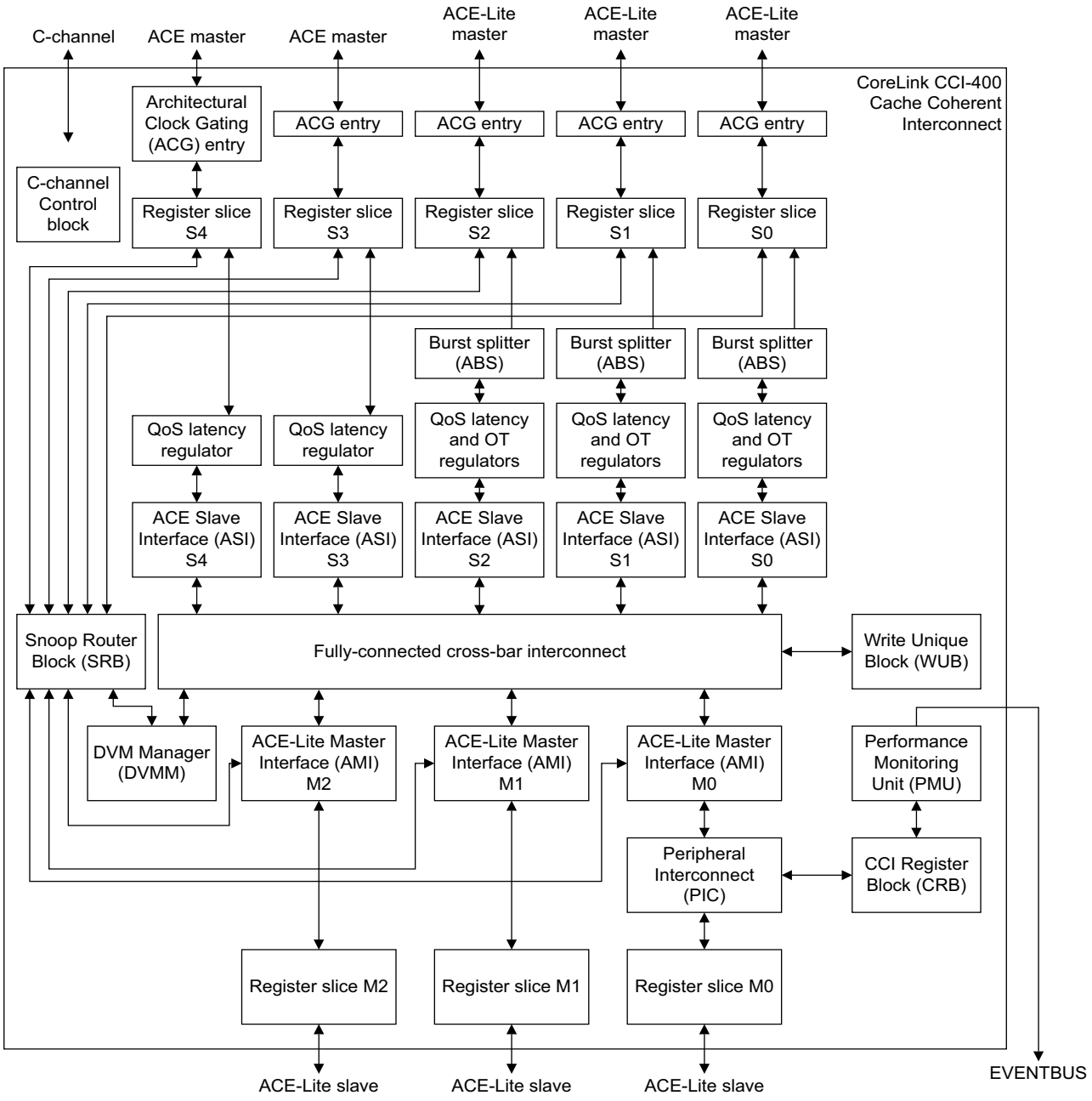

Figure 2-1 CCI-400 internal architecture

2.2 Snoop connectivity and control

The CCI-400 has a fully-connected snoop topology, so if they are enabled:

- each ACE slave interface snoops the other ACE slave interface
- ACE-Lite slave interfaces snoop both ACE slave interfaces
- DVM messages are broadcast through all enabled slave interfaces.

Snooping and DVM message broadcast are disabled at reset, so you must enable the appropriate masters for snooping and DVM messages using the Snoop Control Registers before shareable or DVM messages are sent to the CCI-400. See *Snoop Control Registers* on page 3-12.

2.3 Speculative fetches

For an application where the probability of a miss is high, then the snoop request and response time adds directly to the latency for each transaction labelled as shareable. To mitigate this, you can program each master interface to issue a fetch downstream in parallel with issuing a snoop request. This is known as a *speculative fetch*.

In the event that the snoop associated with a speculative fetch hits in a cache, then the data from the snoop is returned in preference to the data from the speculative fetch.

A speculative fetch bypasses the PoS, and therefore, might be made invalid by a transaction that should have been ordered before it. If a hazardous transaction is detected, or the data is returned before the snoop response is received, then the data from the speculative fetch is discarded and the request retried when the snoop response is received.

You can use the PMU to record the number of retry transactions for each master interface. See *Performance Monitoring Unit (PMU)* on page 2-5.

Note

Speculative fetches are only issued for read-type transactions, that is, `ReadOnce`, `ReadClean`, `ReadNotSharedDirty`, `ReadUnique`, and `ReadShared`.

Although speculative fetches reduce the latency in the case of a snoop miss, there is a bandwidth and power penalty because of the additional transactions on a snoop hit. Therefore, only enable speculative fetches where you expect most snoops to miss. You can use the Speculation Control Register to disable speculative fetches. See *Speculation Control Register* on page 3-7.

2.4 Performance Monitoring Unit (PMU)

The *Performance Monitoring Unit* (PMU) gathers statistics on the performance and behavior of the CCI-400 and attached masters. The PMU consists of:

- a 121-bit event bus, EVNTBUS, that you can export from the CCI-400
- four 32-bit event counters, that you can program to count one event from the event bus
- a clock cycle counter, CCNT, for calculating timing information from the event counters
- an input signal, **NIDEN**, that if HIGH, enables the counting and exporting of events
- an input signal, **SPNIDEN**, that if HIGH, enables the counting of both non-secure and secure events
- a set of counter overflow outputs, **nEVNTCNTOVERFLOW**, that can raise an interrupt when a number of events have occurred.

This section describes:

- *Event list*
- *PMU registers* on page 2-8
- *Using the PMU* on page 2-8.

2.4.1 Event list

Each event has an 8-bit source code, made up as `{source,event}`. Each source is allocated a 3-bit code. Table 2-1 shows the 3-bit source code for events.

Table 2-1 3-bit source code for events

Code[7:5]	Source
`0x0`	Slave interface 0, S0
`0x1`	Slave interface 1, S1
`0x2`	Slave interface 2, S2
`0x3`	Slave interface 3, S3
`0x4`	Slave interface 4, S4
`0x5`	Master interface 0, M0
`0x6`	Master interface 1, M1
`0x7`	Master interface 2, M2

The event list that Table 2-2 on page 2-6 shows consists of a number of events from slave interfaces and different events from master interfaces.

By default, only events relating to non-secure transactions are recorded. However, if the **SPNIDEN** input is HIGH, then both secure and non-secure events are counted and exported.

Note

Stall events are not labeled with a security indicator, so are counted irrespective of the **SPNIDEN** input.

Table 2-2 shows the 5-bit event code event list.

Table 2-2 5-bit event code event list

Code[4:0]	Event	Valid sources
0x00	Read request handshake: any.	Slave interface, S0-S4
0x01	Read request handshake: device transaction.	
0x02	Read request handshake: normal, non-shareable or system-shareable, but not barrier or cache maintenance operation.	
0x03	Read request handshake: inner- or outer-shareable, but not barrier, DVM message or cache maintenance operation.	
0x04	Read request handshake: cache maintenance operation, `CleanInvalid`, `CleanShared`, `MakeInvalid`.	
0x05	Read request handshake: memory barrier.	
0x06	Read request handshake: synchronization barrier.	
0x07	Read request handshake: DVM message, not synchronization.	
0x08	Read request handshake: DVM message, synchronization.	
0x09	Read request stall cycle because the transaction tracker is full. Increase SIx_R_MAX to avoid this stall.	
0x0A	Read data last handshake: data returned from the snoop instead of from downstream.	
0x0B	Read data stall cycle: **RVALIDS** is HIGH, **RREADYS** is LOW.	
0x0C	Write request handshake: any.	
0x0D	Write request handshake: device transaction.	
0x0E	Write request handshake: normal, non-shareable, or system-shareable, but not barrier.	
0x0F	Write request handshake: inner- or outer-shareable, `WriteBack` or `WriteClean`.	
0x10	Write request handshake: `WriteUnique`.	
0x11	Write request handshake: `WriteLineUnique`.	
0x12	Write request handshake: `Evict`.	
0x13	Write request stall cycle because the transaction tracker is full. Increase SIx_W_MAX to avoid this stall.	
0x14	RETRY of speculative fetch transaction.	Master interface, M0-M2
0x15	Read request stall cycle because of an address hazard.	
0x16	Read request stall cycle because of an ID hazard.	
0x17	Read request stall cycle because the transaction tracker is full. Increase MIx_R_MAX to avoid this stall. See the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual*.	
0x18	Read request stall cycle because of a barrier hazard.	
0x19	Write request stall cycle because of a barrier hazard.	
0x1A	Write request stall cycle because the transaction tracker is full. Increase MIx_W_MAX to avoid this stall. See the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual*.	

Event bus

The CCI-400 exports a vector of event signals, **EVNTBUS**, with the bit allocation that Table 2-3 shows.

Table 2-3 EVNTBUS bit allocation

Bits	Source
[120:114]	AMI2 event bus
[113:107]	AMI1 event bus
[106:100]	AMI0 event bus
[99:80]	ASI4 event bus
[79:60]	ASI3 event bus
[59:40]	ASI2 event bus
[39:20]	ASI1 event bus
[19:0]	ASI0 event bus

Table 2-4 shows the bit positions that the ASI events have within each group of signals on the event bus.

Table 2-4 ASIx event bus

Offset	Event
[19]	Write request stall cycle because the transaction tracker is full. Increase SIx_W_MAX to avoid this stall.
[18]	Write request handshake: `Evict`.
[17]	Write request handshake: `WriteLineUnique`.
[16]	Write request handshake: `WriteUnique`.
[15]	Write request handshake: inner- or outer-shareable, `WriteBack` or `WriteClean`.
[14]	Write request handshake: normal, non-shareable, or system-shareable, but not barrier.
[13]	Write request handshake: device transaction.
[12]	Write request handshake: any.
[11]	Read data stall cycle: **RVALIDS** is HIGH, **RREADYS** is LOW.
[10]	Read data last handshake: data returned from the snoop instead of from downstream.
[9]	Read request stall cycle because the transaction tracker is full. Increase SIx_R_MAX to avoid this stall.
[8]	Read request handshake: DVM message, synchronization.
[7]	Read request handshake: DVM message, non-synchronization.
[6]	Read request handshake: synchronization barrier.
[5]	Read request handshake: memory barrier.
[4]	Read request handshake: cache maintenance operation, `CleanInvalid`, `CleanShared`, `MakeInvalid`.
[3]	Read request handshake, inner- or outer-shareable, but not barrier, DVM message or cache maintenance operation.

Table 2-4 ASIx event bus (continued)

Offset	Event
[2]	Read request handshake: normal, non-shareable or system-shareable, but not barrier or cache maintenance operation.
[1]	Read request handshake: device transaction.
[0]	Read request handshake: any.

Table 2-5 shows the allocation of master interface events.

Table 2-5 AMIx event bus

Offset	Event
[6]	Write request stall cycle, transaction tracker full
[5]	Write request stall cycle, barrier hazard
[4]	Read request stall cycle, barrier hazard
[3]	Read request stall cycle, transaction tracker full
[2]	Read request stall cycle, ID hazard
[1]	Read request stall cycle, address hazard
[0]	Retry of speculative fetch transaction

2.4.2 PMU registers

Chapter 3 *Programmers Model* describes the following registers:

- *Event Select Register (ESR)* on page 3-18
- *Event and Cycle Count Registers* on page 3-18
- *Counter Control Registers* on page 3-18.

2.4.3 Using the PMU

An example of how you can use the PMU is to measure the snoop hit rate for read requests on a particular slave interface. Event `0x03` counts the number of cacheable, shareable read requests and event `0x0A` counts the number of snoop hits. The hit rate is then `0x0A` / `0x03`.

2.5 Security

If you are building a system based on the secure and non-secure capabilities that TrustZone technology provides, then you must consider security issues. This section describes:

- *Internal programmers view*
- *Non-TrustZone-aware masters made secure*
- *Security of master interfaces*
- *Security and the Performance Monitoring Unit (PMU)*.

2.5.1 Internal programmers view

With the exception of the PMU registers, the programmers view defaults to secure access only, as follows:

- Non-secure read requests to secure registers receive a DECERR response, **RRESP[1:0]** == `0b11`, and zeroed data.
- Non-secure write requests to secure registers receive a DECERR response, **BRESP[1:0]** == `0b11` and are *Write-Ignored* (WI).

Note

Some accesses might receive a response before they reach the CCI-400 registers and so do not receive a DECERR response nor affect the register values. An example of this is a cache maintenance operation that incorrectly addresses the CCI-400 register space.

You can override these security settings in the Secure Access Register. At reset, you can only access this using secure requests, but if you write to it, this enables non-secure access to all registers in the CCI-400 except for the Control Override Register and Secure Access Register. See *Control Override Register* on page 3-7 and *Secure Access Register* on page 3-8.

2.5.2 Non-TrustZone-aware masters made secure

A master might require access to the CCI-400 registers, and in this case, you can tie the security transaction indicator bits, **ARPROT[1]** and **AWPROT[1]** LOW, so that all accesses by that master are indicated as secure. This places that master permanently in the secure domain. However, depending on the other usage of that master, this might mean that the overall system is not as secure under all circumstances.

2.5.3 Security of master interfaces

Transactions from the CCI-400 master interfaces always retain the security setting of the originating transactions. This applies to:

- non-shareable transactions
- snoop misses
- speculative fetches
- CCI-400-generated writes.

2.5.4 Security and the Performance Monitoring Unit (PMU)

You can use both secure and non-secure transactions to access the Performance Monitor Unit registers. However, you can configure the PMU to count only non-secure events, or both secure and non-secure events, depending on the **SPNIDEN** input. The default is non-secure events.

If the **SPNIDEN** input is taken HIGH, there is a potential security risk because non-secure software can observe security activity through the performance counters. See Appendix A *Signal Descriptions*. ARM advises that you consider the security to be breached for devices placed in this state and that you take appropriate action.

If the **SPNIDEN** input goes from HIGH to LOW, that is, the PMUs go from counting all events to counting only non-secure events, the counters could contain information relating to secure transactions.Therefore, the software should zero the counters if access to that information could represent a potential security risk.

Note

Unlike ARM CPUs, **SPNIDEN** applies to events from both User and Privileged transactions and the CCI-400 makes no distinction between them.

2.6 Error responses

The CCI-400 uses a mixture of precise and imprecise signaling of error responses, where:

- Precise errors are signalled back to the master on the R and B channels for the precise transaction that caused the error.
- Imprecise errors are not signalled on R and B channels but are instead signalled using the **nERRORIRQ** output pin. You can identify the interface that received the error response by reading the Imprecise Error Register. See *Imprecise Error Register* on page 3-9.

2.6.1 Imprecise errors

Table 2-6 shows the errors that are signalled as imprecise. All other sources of error are signalled precisely.

Note

An error is signalled either precisely or imprecisely, but never both.

Table 2-6 shows the imprecise errors.

Table 2-6 Imprecise errors

Transaction causing error	Channel receiving error	Imprecise error indicator from
A ReadX snoop that misses in the cache and fetches data from downstream	CR	Slave interface receiving the CR response
Distributed Virtual Memory message	CR	Slave interface receiving the CR response
Speculative fetch that must be retried	R	Master interface receiving the R response
Write that the CCI-400 generated	B	Master interface receiving the B response
Snoop error or write error for `WriteUnique` or `WriteLineUnique` transactions that has been split, but not the last transaction in the split sequence	CR or B	Master interface targeted by WU or WLU

The CCI-400 generates a precise DECERR response in the case of a security violation on a CCI-400 register access. See *Imprecise Error Register* on page 3-9 and *Security* on page 2-9.

2.6.2 Errors for `WriteUnique` and `WriteLineUnique`

For WU and WLU transactions, the CCI-400 generates a snoop, `CleanInvalid` or `MakeInvalid`, followed by a `WriteNoSnoop` transaction. If an error response is received for either of these transactions, the original write is responded to with an error. Table 2-7 shows the mapping for errors for `WriteUnique` and `WriteLineUnique`.

Table 2-7 Errors for WriteUnique and WriteLineUnique

Snoop response	WriteNoSnoop response	BRESP to originator
OK	OK	OK
OK	SLVERR	SLVERR
OK	DECERR	DECERR

Table 2-7 Errors for WriteUnique and WriteLineUnique (continued)

Snoop response	WriteNoSnoop response	BRESP to originator
ERROR	OK	SLVERR
ERROR	SLVERR	SLVERR
ERROR	DECERR	DECERR

2.7 Cache maintenance operations

The CCI-400 supports the snooping of cache maintenance operations based on the Snoop Control Register. See *Snoop Control Registers* on page 3-12. You can optionally send cache maintenance operations downstream, depending on the **BROADCASTCACHEMAINT** inputs. See the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* for more information.

2.8 Barriers

The CCI-400 supports all types of AMBA 4 barrier transactions. Each slave interface broadcasts these to each master interface, ensuring that intermediate transaction source and sink points observe the barrier correctly. For more information on barriers, see the *Barriers* section in the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual*.

2.9 Exclusive accesses

The CCI-400 supports exclusive accesses to shareable and non-shareable locations as the ACE and AXI4 protocols describe. The *AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite* permits shareable exclusive accesses on ACE interfaces only.

See the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* for more information on exclusive accesses.

2.10 Distributed Virtual Memory (DVM) messages

The ACE slave interfaces support *Distributed Virtual Memory* (DVM) messages through their regular AC and CR channels. The ACE-Lite interfaces all contain AC and CR channels to support DVM messages only. Each slave interface has a programmable enable bit to determine whether it supports the issuing of AC requests for DVM messages. DVM messages are handled as regular transactions in the CCI-400, except that they are decoded based on the DVM message indicator, instead of the address, to ensure that multi-transaction DVM messages are correctly ordered.

The Snoop Control Registers and Control Override Register control the issuing of DVM message requests. See *Snoop Control Registers* on page 3-12 and *Control Override Register* on page 3-7.

Note

A master that issues DVM messages must also be able to receive DVM messages. The slave interface through which the master connects must have DVM messages enabled.

2.11 Quality-of-Service (QoS)

Quality of Service (QoS) provision in the CCI-400 is achieved using the following independent mechanisms:

- *QoS value*
- *Regulation based on outstanding transactions* on page 2-18.

2.11.1 QoS value

Each CCI-400 slave interface has **ARQOS** and **AWQOS** input signals that transport a transaction-based QoS value. This determines the relative priority between transactions on that interface, or between interfaces. To gain access to a master interface, each transaction passes through an arbitration point, where it is granted based on the QoS value. Inputs with the same QoS value are arbitrated based on a *Least Recently Granted* (LRG) scheme.

The NIC-400 Network Interconnect and DMC-400 Dynamic Memory Controller also support arbitration based on QoS value, so the CCI-400 propagates the QoS value on its master interfaces. Transactions that the CCI-400 generates inherit the QoS value from the instigating transaction.

Note

Ensure that you balance the relative priorities of all slave interfaces. For example, setting each to the highest QoS value reduces the arbitration to LRG and no advantage is gained from having a QoS value.

You can override the **ARQOS** and **AWQOS** input signals from each slave interface by using a programmable register if the relevant static input signal, **QOSOVERRIDE[4:0]**, with one bit for each of slave interfaces 4-0, is HIGH. The QoS override is either based on a programmable value, or uses latency feedback to set the value within a programmable range.

QoS value based on latency measurement

For applications where a fixed QoS value is not flexible enough for a particular slave interface, the CCI-400 provides a mechanism whereby **AxQOS** varies depending on the latency measured at that interface. You can program the range of minimum and maximum priority value for each regulator to permit prioritization under worst case conditions. In addition, you can control the rate of change of the regulator by using a programmable scaling factor, Ki.

When you enable latency regulation, the read and write **AxQOS** values are driven by those generated by the latency monitors. When you disable latency regulation, the input or static values are used, depending on the QoS Override register. The target slave, for example, the DDR controller, should be sensitive to the **AxQOS** value. That is, a higher priority gives a lower latency.

Note

Turning latency regulation on when **QOSOVERRIDE[x]** is set LOW for a specific interface has no effect. The **AxQOS** signals are overridden when **QOSOVERRIDE[x]** is HIGH.

2.11.2 Regulation based on outstanding transactions

The CCI-400 offers an additional mechanism for regulating traffic flows for the benefit of overall performance. Each ACE-Lite slave interface has an optional, programmable, mechanism for limiting the number of outstanding read and write transactions, where an outstanding transaction is a read request without read data, or a write request without a response.

If you enable regulation, the following programmable values set the permitted number of outstanding transactions:

- OT integer
- OT fraction.

You can use these parts to set a fractional number of outstanding transactions, for example, 2.75.

You can program each interface individually, and you can use this regulation to manage the available bandwidth between masters.

Note

- **QOSOVERRIDE[x]** has no effect on outstanding transaction rate regulation.
- Only transactions that have data associated with them are counted for QoS regulation. These transactions are:
  - `ReadNoSnoop`
  - `ReadOnce`
  - `ReadShared`
  - `ReadClean`
  - `ReadNotSharedDirty`
  - `ReadUnique`
  - `WriteNoSnoop`
  - `WriteUnique`
  - `WriteLineUnique`
  - `WriteClean`
  - `WriteBack`.

2.11.3 QoS programmable registers

Chapter 3 *Programmers Model* describes the following registers:

- *Read Channel QoS Value Override Register* on page 3-14
- *Write Channel QoS Value Override Register* on page 3-14
- *QoS Control Register* on page 3-14
- *Max OT Registers* on page 3-15
- *Target Latency Registers* on page 3-16
- *Latency Regulation Registers* on page 3-16.

2.12 Clock and reset

This section describes:

- *Clocking*
- *Reset*.

2.12.1 Clocking

The CCI-400 has a single main clock, **ACLK**, that is distributed to all sub-blocks. Where masters and slaves connecting to the CCI-400 are in a different clock domain, it is necessary to use external clock-domain-crossing bridges. For more information on hierarchical clock gating, see the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* and *CoreLink CCI-400 Cache Coherent Interconnect Implementation Guide*.

2.12.2 Reset

The CCI-400 has a single reset domain with an active LOW reset input signal, **ARESETn**. This is synchronized to **ACLK** with a double-register on the input to the CCI-400.

Note

There must be no activity on the slave interfaces, and the configuration inputs must be static for at least three **ACLK** cycles after **ARESETn** goes HIGH.

Chapter 3
Programmers Model

This chapter describes the programmers model.

It contains the following sections:

3.1 About this programmers model

The following information applies to the CCI-400 Cache Coherent Interconnect registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in unpredictable behavior.
- Unless otherwise stated in the accompanying text:
  - do not modify undefined register bits
  - ignore undefined register bits on reads
  - all register bits are reset to a logic 0 by a system or power-on reset.
- Table 3-1 on page 3-3 describes access types as follows:

R/W Read and write.

RAZ Read as zero.

WI Write ignored.

The CCI-400 registers occupy a 64KB region and are offset using the **PERIPHBASE[39:15]** static input.

The following rules apply:

- You can access the PV using only Device-type transactions, **AxCACHE[1]** = 0, of length 1 and up to 32 bits in size. Transactions that do not meet these requirements and reach the CCI-400 register region receive a DECERR response.
- A transaction to a reserved block, or a transaction that violates security restrictions, receives a DECERR response.

The programmers view contains regions for control, slave interface, and performance counter registers. See Table 3-1 on page 3-3.

3.2 Register summary

Table 3-1 shows the registers in offset order from the base memory address, **PERIPHBASE[39:15]**.

Note

The base address for internal CCI-400 registers is defined using a static input, **PERIPHBASE[39:15]**. The CCI-400 registers are offset by `0x90000` from this base address and occupy an address region of size 64KB.

For example, if PERIPHBASE is `0x0000000`, then the register space occupies the region `0x0000090000` to `0x000009FFFF`.

Table 3-1 Register summary

Offset	Type	Reset	Width	Description
`0x90000`	R/W	`0x0`	32	*Control Override Register* on page 3-7.
`0x90004`	R/W	`0x0`	32	*Speculation Control Register* on page 3-7.
`0x90008`	R/W	`0x0`	32	*Secure Access Register* on page 3-8.
`0x9000C`	R	`0x0`	32	*Status Register* on page 3-8.
`0x90010`	R/W	`0x0`	32	*Imprecise Error Register* on page 3-9.
`0x90100`	R/W	`0x2000`	32	*Performance Monitor Control Register (PMCR)* on page 3-10.
`0x90FD0 - 0x90FFC`	R/W	`0x0`	32	*Component and Peripheral ID Registers* on page 3-11.
Slave interface 0 registers				
`0x91000`	R/W	`0x0`	32	Snoop Control Register for slave interface 0. See *Snoop Control Registers* on page 3-12.
`0x91004`	R/W	`0x0`	32	Shareable Override Register for slave interface 0. See *Shareable Override Register* on page 3-13.
`0x91100`	R/W	`0x0`	32	Read Channel QoS Value Override Register for slave interface 0. See *Read Channel QoS Value Override Register* on page 3-14.
`0x91104`	R/W	`0x0`	32	Write Channel QoS Value Override slave interface 0. See *Write Channel QoS Value Override Register* on page 3-14.
`0x9110C`	R/W	`0x0`	32	QoS Control Register for slave interface 0. See *QoS Control Register* on page 3-14.
`0x91110`	R/W	`0x0`	32	Max OT Register for slave interface 0. See *Max OT Registers* on page 3-15.
`0x91130`	R/W	`0x0`	32	Target Latency Register for slave interface 0. See *Target Latency Registers* on page 3-16.
`0x91134`	R/W	`0x0`	32	Latency Regulation Register for slave interface 0. See *Latency Regulation Registers* on page 3-16.
`0x91138`	R/W	`0x0`	32	QoS Range Register for slave interface 0. See *QoS Range Register* on page 3-17.

Table 3-1 Register summary (continued)

Offset	Type	Reset	Width	Description
Slave interface 1 registers				
0x92000	R/W	0x0	32	Snoop Control Register for slave interface 1. See *Snoop Control Registers* on page 3-12.
0x92004	R/W	0x0	32	Shareable Override Register for slave interface 1. See *Shareable Override Register* on page 3-13.
0x92100	R/W	0x0	32	Read Channel QoS Value Override Register for slave interface 1. See *Read Channel QoS Value Override Register* on page 3-14.
0x92104	R/W	0x0	32	Write Channel QoS Value Override slave interface 1. See *Write Channel QoS Value Override Register* on page 3-14.
0x9210C	R/W	0x0	32	QoS Control Register for slave interface 1. See *QoS Control Register* on page 3-14.
0x92110	R/W	0x0	32	Max OT Register for slave interface 1. See *Max OT Registers* on page 3-15.
0x92130	R/W	0x0	32	Target Latency Register for slave interface 1. See *Target Latency Registers* on page 3-16.
0x92134	R/W	0x0	32	Latency Regulation Register for slave interface 1. See *Latency Regulation Registers* on page 3-16.
0x92138	R/W	0x0	32	QoS Range Register for slave interface 1. See *QoS Range Register* on page 3-17.
Slave interface 2 registers				
0x93000	R/W	0x0	32	Snoop Control Register for slave interface 2. See *Snoop Control Registers* on page 3-12.
0x93004	R/W	0x0	32	Shareable Override Register for slave interface 2. See *Shareable Override Register* on page 3-13.
0x93100	R/W	0x0	32	Read Channel QoS Value Override Register for slave interface 2. See *Read Channel QoS Value Override Register* on page 3-14.
0x93104	R/W	0x0	32	Write Channel QoS Value Override slave interface 2. See *Write Channel QoS Value Override Register* on page 3-14.
0x9310C	R/W	0x0	32	QoS Control Register for slave interface 2. See *QoS Control Register* on page 3-14.
0x93110	R/W	0x0	32	Max OT Register for slave interface 2. See *Max OT Registers* on page 3-15.
0x93130	R/W	0x0	32	Target Latency Register for slave interface 2. See *Target Latency Registers* on page 3-16.
0x93134	R/W	0x0	32	Latency Regulation Register for slave interface 2. See *Latency Regulation Registers* on page 3-16.
0x93138	R/W	0x0	32	QoS Range Register for slave interface 2. See *QoS Range Register* on page 3-17.

Table 3-1 Register summary (continued)

Offset	Type	Reset	Width	Description
Slave interface 3 registers				
0x94000	R/W	0x0	32	Snoop Control Register for slave interface 3. See *Snoop Control Registers* on page 3-12.
0x94100	R/W	0x0	32	Read Channel QoS Value Override Register for slave interface 3. See *Read Channel QoS Value Override Register* on page 3-14.
0x94104	R/W	0x0	32	Write Channel QoS Value Override Register for slave interface 3. See *Write Channel QoS Value Override Register* on page 3-14.
0x9410C	R/W	0x0	32	QoS Control Register for slave interface 3. See *QoS Control Register* on page 3-14.
0x94130	R/W	0x0	32	Target Latency Register for slave interface 3. See *Target Latency Registers* on page 3-16.
0x94134	R/W	0x0	32	Latency Regulation Register for slave interface 3. See *Latency Regulation Registers* on page 3-16.
0x94138	R/W	0x0	32	QoS Range Register for slave interface 3. See *QoS Range Register* on page 3-17.
Slave interface 4 registers				
0x95000	R/W	0x0	32	Snoop Control Register for slave interface 4. See *Snoop Control Registers* on page 3-12.
0x95100	R/W	0x0	32	Read Channel QoS Value Override Register for slave interface 4. See *Read Channel QoS Value Override Register* on page 3-14.
0x95104	R/W	0x0	32	Write Channel QoS Value Override slave interface 4. See *Write Channel QoS Value Override Register* on page 3-14.
0x9510C	R/W	0x0	32	QoS Control Register for slave interface 4. See *QoS Control Register* on page 3-14.
0x95130	R/W	0x0	32	Target Latency Register for slave interface 4. See *Target Latency Registers* on page 3-16.
0x95134	R/W	0x0	32	Latency Regulation Register for slave interface 4. See *Latency Regulation Registers* on page 3-16.
0x95138	R/W	0x0	32	QoS Range Register for slave interface 4. See *QoS Range Register* on page 3-17.
Cycle counter registers				
0x99004	R/W	0x0	32	Cycle counter register. See *Event and Cycle Count Registers* on page 3-18.
0x99008	R/W	0x0	32	Count Control Register for cycle counter. See *Counter Control Registers* on page 3-18.
0x9900C	R/W	0x0	32	Overflow Flag Status Register for cycle counter. See *Overflow Flag Status Register* on page 3-19.

Table 3-1 Register summary (continued)

Offset	Type	Reset	Width	Description
Performance counter 0 registers				
0x9A000	R/W	0x0	32	Event Select Register for performance counter 0. See *Event Select Register (ESR)* on page 3-18.
0x9A004	R/W	0x0	32	Event Count Register for performance counter 0. See *Event and Cycle Count Registers* on page 3-18.
0x9A008	R/W	0x0	32	Counter Control Register for performance counter 0. See *Counter Control Registers* on page 3-18.
0x9A00C	R/W	0x0	32	Overflow Flag Status Register for performance counter 0. See *Overflow Flag Status Register* on page 3-19.
Performance counter 1 registers				
0x9B000	R/W	0x0	32	Event Select Register for performance counter 1. See *Event Select Register (ESR)* on page 3-18.
0x9B004	R/W	0x0	32	Event Count Register for performance counter 1. See *Event and Cycle Count Registers* on page 3-18.
0x9B008	R/W	0x0	32	Counter Control Register for performance counter 1. See *Counter Control Registers* on page 3-18.
0x9B00C	R/W	0x0	32	Overflow Flag Status Register for performance counter 1. See *Overflow Flag Status Register* on page 3-19.
Performance counter 2 registers				
0x9C000	R/W	0x0	32	Event Select Register for performance counter 2. See *Event Select Register (ESR)* on page 3-18.
0x9C004	R/W	0x0	32	Event Count Register for performance counter 2. See *Event and Cycle Count Registers* on page 3-18.
0x9C008	R/W	0x0	32	Counter Control Register for performance counter 2. See *Counter Control Registers* on page 3-18.
0x9C00C	R/W	0x0	32	Overflow Flag Status Register for performance counter 2. See *Overflow Flag Status Register* on page 3-19.
Performance counter 3 registers				
0x9D000	R/W	0x0	32	Event Select Register for performance counter 3. See *Event Select Register (ESR)* on page 3-18.
0x9D004	R/W	0x0	32	Event Count Register for performance counter 3. See *Event and Cycle Count Registers* on page 3-18.
0x9D008	R/W	0x0	32	Counter Control Register for performance counter 3. See *Counter Control Registers* on page 3-18.
0x9D00C	R/W	0x0	32	Overflow Flag Status Register for performance counter 3. See *Overflow Flag Status Register* on page 3-19.
0x9E000 - 0x9FFFF	R/W	0x0	32	Reserved

3.3 Register descriptions

This section describes the CCI-400 registers. Table 3-1 on page 3-3 provides cross references to individual registers.

3.3.1 Control Override Register

The Control Override Register characteristics are:

Purpose Additional control register that provides a fail-safe override for some CCI-400 functions, if these cause problems that you cannot otherwise work around.

Usage constraints If you cannot avoid using them, only set them using non-bufferable transactions, and before barriers, shareable transactions, or DVM messages are issued into the CCI-400. This could be, for example, very early in the boot sequence, prior to the installation of any secure OS. You can access the Control Override Register using secure transactions only, irrespective of the programming of the Secure Access Register.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-2 shows the bit assignments.

Table 3-2 Control Override Register

Bits	Reset value	Access	Function
[31:4]	-	RAZ/WI	Reserved
[3]	0x0	R/W	Terminate all barrier transactions. The options are as follows: 0b0 Master interfaces terminate barriers according to the **BARRIERTERMINATE** inputs. 0b1 All master interfaces terminate barriers.
[2]	0x0	R/W	Disable speculative fetches. The options are as follows: 0b0 Send speculative fetches according to the Speculation Control Register. See *Speculation Control Register*. 0b1 Disable speculative fetches from all master interfaces.
[1]	0x0	R/W	DVM message disable. The options are as follows: 0b0 Send DVM messages according to the Snoop Control Registers. See *Snoop Control Registers* on page 3-12. 0b1 Disable propagation of all DVM messages.
[0]	0x0	R/W	Snoop disable. The options are as follows: 0b0 Snoop masters according to the Snoop Control Registers. See *Snoop Control Registers* on page 3-12. 0b1 Disable all snoops, but not DVM messages.

3.3.2 Speculation Control Register

The Speculation Control Register characteristics are:

Purpose Disables speculative fetches for a master interface.

Usage constraints Access controlled by Secure Access Register, see *Secure Access Register* on page 3-8.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-3 shows the bit assignments.

Table 3-3 Speculation Control Register

Bits	Reset value	Access	Function
[31:3]	-	RAZ/WI	Reserved
[2:0]	0x0	R/W	Disable speculative fetches from a master interface. One bit for each master interface, M2, M1, M0: 0b0 Enable speculative fetches. 0b1 Disable speculative fetches.

3.3.3 Secure Access Register

The Secure Access Register characteristics are:

Purpose Controls secure access.

Usage constraints You can only write to this register using secure transactions.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Warning

This register enables non-secure access to the CCI-400 registers for all masters. This compromises the security of your system.

Table 3-4 shows the bit assignments.

Table 3-4 Secure Access Register

Bits	Reset value	Access	Function
[31:1]	-	RAZ/WI	Reserved
[0]	0x0	R/W	Non-secure register access override. 0b0 Disable non-secure access to CCI-400 registers. 0b1 Enable non-secure access to CCI-400 registers.

3.3.4 Status Register

The Status Register characteristics are:

Purpose Safely enables and disables snooping. When changing the snoop or DVM message enables using the Snoop Control Registers, see *Snoop Control Registers* on page 3-12, there is a delay until the changes are registered in all parts of the CCI-400. The `change_pending` bit in the Status Register indicates whether there are any changes to the enables that have not yet been applied, or whether a slave interface has been disabled for future snoop and DVM messages, but has outstanding AC requests.

Note

After changing the snoop and DVM enable bits, the controller must test that the change_pending bit is LOW before it turns an attached device on or off.

Usage constraints There are no usage constraints.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-5 shows the bit assignments.

Table 3-5 Status Register

Bits	Reset value	Access	Function
[31:1]	-	RAZ/WI	Reserved
[0]	0x0	R/WI	Indicates whether any changes to the snoop or DVM enables is pending in the CCI-400: 0b0 No change pending. 0b1 Change pending.

3.3.5 Imprecise Error Register

The Imprecise Error Register characteristics are:

Purpose Records the CCI-400 interfaces that received an error that is not signalled precisely. The appropriate bit is set, with respect to the interface on which the error was received. Bits are set when one or more error responses are detected, and they are reset on a write of 1 to the corresponding bit.

Note

If any of the imprecise error indicator bits are set, the **nERRORIRQ** signal is asserted, active LOW.

Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Usage constraints There are no usage constraints.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-6 shows the bit assignments.

Table 3-6 Imprecise Error Register

Bits	Reset value	Access	Function
[31:21]	-	RAZ/WI	Reserved
[20]	0x0	R/W	Imprecise error indicator for slave interface 4
[19]	0x0	R/W	Imprecise error indicator for slave interface 3
[18]	0x0	R/W	Imprecise error indicator for slave interface 2
[17]	0x0	R/W	Imprecise error indicator for slave interface 1

Table 3-6 Imprecise Error Register (continued)

Bits	Reset value	Access	Function
[16]	0x0	R/W	Imprecise error indicator for slave interface 0
[15:3]	-	RAZ/WI	Reserved
[2]	0x0	R/W	Imprecise error indicator for master interface 2
[1]	0x0	R/W	Imprecise error indicator for master interface 1
[0]	0x0	R/W	Imprecise error indicator for master interface 0: 0b0 No error since this bit was last reset. 0b1 An error response has been received, but not signalled precisely.

3.3.6 Performance Monitor Control Register (PMCR)

The *Performance Monitor Control Register* (PMCR) characteristics are:

Purpose Controls the performance monitor.

Usage constraints There are no usage constraints.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-7 shows the bit assignments.

Table 3-7 Performance Monitor Control Register

Bits	Name	Reset Value	Access	Function
[31:16]	-	-	RAZ/WI	Reserved
[15:11]	-	0x4	R/WI	Specifies the number of counters implemented
[10:6]	-	-	RAZ/WI	Reserved
[5]	DP	0x0	R/W	Disables cycle counter, CCNT, if non-invasive debug is prohibited. The options are as follows: 0b0 Count is not disabled when **NIDEN** input is LOW. 0b1 Count is disabled when **NIDEN** input is LOW.
[4]	EX	0x0	R/W	Enable export of the events to the event bus, **EVNTBUS**, for an external monitoring block to trace events. The options are as follows: 0b0 Do not export **EVNTBUS**. 0b1 Export **EVNTBUS**.
[3]	CCD	0x0	R/W	Cycle count divider. The options are as follows: 0b0 Count every clock cycle when enabled. 0b1 Count every 64th clock cycle when enabled.

Table 3-7 Performance Monitor Control Register (continued)

Bits	Name	Reset Value	Access	Function
[2]	CCR	0x0	RAZ/W	Cycle counter reset. The options are as follows: 0b0 No action. 0b1 Reset cycle counter, CCNT, to zero.
[1]	RST	0x0	RAZ/W	Performance counter reset. The options are as follows: 0b0 No action. 0b1 Reset all performance counters to zero, not including CCNT.
[0]	CEN	0x0	R/W	Enable bit. The options are as follows: 0b0 Disable all counters, including CCNT. 0b1 Enable all counters including CCNT.

Table 3-8 shows the relationship between the non-invasive debug enable input, **NIDEN**, and the PMCR register settings.

Table 3-8 Relationship between non-invasive debug enable input, NIDEN, and PMCR register settings

NIDEN input	PMCR.DP	PMCR.EX	Event counters enabled	Events exported	Cycle counter enabled
1	X	0	Yes	No	Yes
1	X	1	Yes	Yes	Yes
0	0	X	No	No	Yes
0	1	X	No	No	No

3.3.7 Component and Peripheral ID Registers

Table 3-9 shows the values for the Component and Peripheral Identification registers for the CCI-400.

In each of these registers, the most significant 24 bits are RAZ/WI. The least significant 8 bits of the four Component ID registers form a single 32-bit conceptual ID register. In a similar way, the defined fields of the eight Peripheral ID registers form a conceptual 64-bit ID register.

Table 3-9 Component and Peripheral ID registers

Register	Offset	Bits	Value	Description
Peripheral ID4	0xFD0	[3:0]	0x4	JEP106 continuation code for ARM.
		[7:4]	0x4	4KB region count.
Peripheral ID5	0xFD4	[7:0]	0x00	Reserved.
Peripheral ID6	0xFD8	[7:0]	0x00	Reserved.
Peripheral ID7	0xFDC	[7:0]	0x00	Reserved.
Peripheral ID0	0xFE0	[7:0]	0x20	Part number[7:0].
Peripheral ID1	0xFE4	[3:0]	0x4	Part number[11:8].
		[7:4]	0xB	JEP106 ID code[3:0] for ARM.

Table 3-9 Component and Peripheral ID registers (continued)

Register	Offset	Bits	Value	Description
Peripheral ID2	0xFE8	[2:0]	0x3	JEP106 ID code[6:4] for ARM.
		[3]	0x1	IC uses a manufacturer's identity code allocated by JEDEC according to the JEP106 specification.
		[7:4]	0x3	CCI-400 revision, r0p3.
Peripheral ID3	0xFEC	[3:0]	0x0	Customer modification number.
		[7:4]	0x0	ARM approved ECO number. Use the **ECOREVNUM** inputs to modify this value.
Component ID0	0xFF0	[7:0]	0x0D	These values identify the CCI-400 as an ARM component.
Component ID1	0xFF4	[7:0]	0xF0	
Component ID2	0xFF8	[7:0]	0x05	
Component ID3	0xFFC	[7:0]	0xB1	

ECO revision number

To track any *Engineering Change Order* (ECO) fixes in the CCI-400, you can change part of the peripheral ID register using the **ECOREVNUM** input pins. You must tie these signals LOW unless you have an ECO from ARM.

Each bit of the **ECOREVNUM** input corresponds to one of bits [7:4] of the Peripheral ID3 register, MSB to MSB. Driving an input HIGH inverts the associated ID3 bit.

Note

ARM recommends you ensure that each of the signal drivers is distinct and readily identifiable to facilitate possible metal layer modification.

3.3.8 Snoop Control Registers

One Snoop Control Register exists for each slave interface. Table 3-10 shows the bit assignments.

Table 3-10 Snoop Control Registers

Bits	Reset value	Access	Function
[31]	**ACCHANNELEN** input	R/WI	Slave interface supports DVM messages. This is overridden to 0x0 if you set the Control Override Register [1]. See *Control Override Register* on page 3-7.
[30]	**ACCHANNELEN** input for S3 and S4 0x0 for S0, S1, and S2	R/WI	Slave interface supports snoops. This is overridden to 0x0 if you set the Control Override Register [0]. See *Control Override Register* on page 3-7.

Table 3-10 Snoop Control Registers (continued)

Bits	Reset value	Access	Function
[29:2]	-	RAZ/WI	Reserved.
[1]	0x0	R/W	Enable issuing of DVM message requests from this slave interface. RAZ/WI for interfaces not supporting DVM messages: 0b0 Disable DVM message requests. 0b1 Enable DVM message requests.
[0]	0x0	R/W	Enable issuing of snoop requests from this slave interface. RAZ/WI for interfaces not supporting snoops: 0b0 Disable snoop requests. 0b1 Enable snoop requests.

3.3.9 Shareable Override Register

The Shareable Override Register characteristics are:

Purpose Overrides shareability of normal transactions through this interface. The following transaction types are unaffected by any override:

- FIXED-type bursts
- device
- exclusive accesses
- cache maintenance
- barrier
- DVM message transactions.

Usage constraints This register is for ACE-Lite slave interfaces only. Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-11 shows the bit assignments.

Table 3-11 Shareable Override Register

Bits	Reset	Access	Function
[31:2]	-	RAZ/WI	Reserved.
[1:0]	0x0	R/W	Shareable override for slave interface 0x0, 0x1 Do not override **AxDOMAIN** inputs. 0x2 Override **AxDOMAIN** inputs to 0b00, all transactions are treated as non-shareable: • ReadOnce becomes ReadNoSnoop. • WriteUnique and WriteLineUnique become WriteNoSnoop. 0x3 Override **AxDOMAIN** inputs to 0b01, normal transactions are treated as shareable: • ReadNoSnoop becomes ReadOnce. • WriteNoSnoop becomes WriteUnique.

3.3.10 Read Channel QoS Value Override Register

The Read Channel QoS Value Override Register characteristics are:

Purpose Contains override values for **ARQOS**, with a register for each slave interface. These are overridden if you set the relevant **QOSOVERRIDE[4:0]** input signal bit.

Usage constraints Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-12 shows the bit assignments.

Table 3-12 Read Channel QoS Value Override Register

Bits	Reset value	Access	Function
[31:4]	-	RAZ/WI	Reserved
[3:0]	0x0	R/W	**ARQOS** value override for slave interface

3.3.11 Write Channel QoS Value Override Register

The Write Channel QoS Value Override Register characteristics are:

Purpose Contains override values for **AWQOS**, with a register for each slave interface. These are overridden if you set the relevant **QOSOVERRIDE[4:0]** input signal bit.

Usage constraints Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-13 shows the bit assignments.

Table 3-13 Write Channel QoS Value Override Register

Bits	Reset value	Access	Function
[31:4]	-	RAZ/WI	Reserved
[3:0]	0x0	R/W	**AWQOS** value override for slave interface

3.3.12 QoS Control Register

The QoS Control Register characteristics are:

Purpose Controls the regulators that are enabled on the slave interfaces.

Usage constraints Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-14 shows the bit assignments.

Table 3-14 QoS Control Register

Bits	Reset value	Access	Function
[31:4]	-	RAZ/WI	Reserved
[3]	0x0	R/W	Enable regulation of outstanding read transactions for slave interface. • ACE-Lite interfaces only, S0, S1, S2 • RAZ/WI for ACE interfaces, S3 and S4.
[2]	0x0	R/W	Enable regulation of outstanding write transactions for slave interface • ACE-Lite interfaces only, S0, S1, S2 • RAZ/WI for ACE interfaces, S3 and S4.
[1]	0x0	R/W	Enable regulation of AR latency for slave interface
[0]	0x0	R/W	Enable regulation of AW latency for slave interface

3.3.13 Max OT Registers

The Max OT Registers characteristics are:

Purpose Determine how many outstanding transactions are permitted when the OT regulator is enabled for each ACE-Lite slave interface. One register exists for each of the S0, S1, and S2 slave interfaces. A value of 0 for both the integer and fractional parts disables the programmable regulation so that the hardware limits apply. A value of 0 for the fractional part disables the regulation of fractional outstanding transactions. If `int` is the value of the integer part and `frac` is the value of the fractional part, then:

Maximum number of outstanding transactions = `int` + `frac`/256.

Usage constraints Setting the maximum outstanding transaction size greater than that configured in the RTL, using the `R_MAX` or `W_MAX` parameters, has no effect. See the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual* for information on the design-time configuration of the CCI-400. Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-15 shows the bit assignments.

Table 3-15 Max OT Register

Bits	Reset value	Access	Function
[31:30]	-	RAZ/WI	Reserved
[29:24]	0x0	R/W	Integer part of the maximum outstanding AR addresses
[23:16]	0x0	R/W	Fractional part of the maximum outstanding AR addresses
[15:14]	-	RAZ/WI	Reserved
[13:8]	0x0	R/W	Integer part of the maximum outstanding AW addresses
[7:0]	0x0	R/W	Fractional part of the maximum outstanding AW addresses

3.3.14 Target Latency Registers

The Target Latency Registers characteristics are:

Purpose Determine the target latency, in cycles, for the regulation of reads and writes. A value of 0 corresponds to no regulation. One register exists for each slave interface.

Usage constraints Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Only has an effect when **QOSOVERRIDE** is HIGH for the associated interface.

Attributes See Table 3-1 on page 3-3.

Table 3-16 shows the bit assignments.

Table 3-16 Target Latency Register

Bits	Reset value	Access	Function
[31:28]	-	RAZ/WI	Reserved
[27:16]	0x0	R/W	AR channel target latency
[15:12]	-	RAZ/WI	Reserved
[11:0]	0x0	R/W	AW channel target latency

3.3.15 Latency Regulation Registers

The Latency Regulation Registers characteristics are:

Purpose Latency regulation value, **AWQOS** or **ARQOS**, scale factor coded for powers of 2 in the range 2^{-5}-2^{-12}, to match a 16-bit integrator. One register exists for each slave interface.

Usage constraints Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Only has an effect when **QOSOVERRIDE** is HIGH for the associated interface.

Attributes See Table 3-1 on page 3-3.

Table 3-17 shows the bit assignments.

Table 3-17 Latency Regulation Register

Bits	Reset value	Access	Function
[31:11]	-	RAZ/WI	Reserved
[10:8]	0x0	R/W	**ARQOS** scale factor, power of 2 in the range 2^{-5}-2^{-12}
[7:3]	-	RAZ/WI	Reserved
[2:0]	0x0	R/W	**AWQOS** scale factor, power of 2 in the range 2^{-5}-2^{-12}

Table 3-18 shows the mapping of the latency regulation value to the latency regulation scale factor.

Table 3-18 Mapping of latency regulation value to scale factor

Latency regulation value	Latency regulation scale factor
0x0	2^{-5}
0x1	2^{-6}
0x2	2^{-7}
0x3	2^{-8}
0x4	2^{-9}
0x5	2^{-10}
0x6	2^{-11}
0x7	2^{-12}

3.3.16 QoS Range Register

The QoS Range Register characteristics are:

Purpose Enables you to program the minimum and maximum values for the **ARQOS** and **AWQOS** signals that the latency regulators generate. One register exists for each slave interface.

Usage constraints Accessible using only secure accesses, unless you set the Secure Access Register. See *Secure Access Register* on page 3-8.

Configurations Only has an effect when **QOSOVERRIDE** is HIGH for the associated interface.

Attributes See Table 3-1 on page 3-3.

Table 3-19 shows the bit assignments.

Table 3-19 QoS Range Register

Bits	Reset value	Access	Function
[31:28]	-	RAZ/WI	Reserved
[27:24]	0x0	R/W	Maximum ARQOS value
[23:20]	-	RAZ/WI	Reserved
[19:16]	0x0	R/W	Minimum ARQOS value
[15:12]	-	RAZ/WI	Reserved
[11:8]	0x0	R/W	Maximum AWQOS value
[7:4]	-	RAZ/WI	Reserved
[3:0]	0x0	R/W	Minimum AWQOS value

3.3.17 Event Select Register (ESR)

The Event Select Register characteristics are:

Purpose Determines the event that a particular counter counts. One register exists per counter. This register is not present for the cycle counter.

Usage constraints There are no usage constraints.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-20 shows the bit assignments.

Table 3-20 Event Select Register

Bits	Reset value	Access	Function
[31:8]	-	RAZ/WI	Reserved.
[7:0]	0x0	R/W	Event number for counter. See *Event list* on page 2-5.

3.3.18 Event and Cycle Count Registers

A read-write, 32-bit register for each of the four event counters and cycle counter. The cycle counter counts either every CCI-400 clock cycle, or every 64 clock cycles, depending on the **PMCR** bit [3].

You can reset all event counter values to zero by writing a 1 to the **PMCR** bit [1] and reset all clock counter values by writing a 1 to **PMCR** bit [2].

3.3.19 Counter Control Registers

The Counter Control Registers characteristics are:

Purpose Enable or disable the Cycle and Event Counters. One register exists per counter.

Usage constraints There are no usage constraints.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-21 shows the bit assignments.

Table 3-21 Counter Control Register bit assignments

Bits	Reset value	Access	Function
[31:1]	-	RAZ/WI	Reserved
[0]	0x0	R/W	Counter enable. The options are as follows: 0b0 Counter disabled. 0b1 Counter enabled.

3.3.20 Overflow Flag Status Register

The Overflow Flag Status Register characteristics are:

Purpose Contains the state of the overflow flags for the Cycle Count Register and event counters. One register exists for each event counter, and one register exists for the cycle counter.

Usage constraints There are no usage constraints.

Configurations Available in all CCI-400 configurations.

Attributes See Table 3-1 on page 3-3.

Table 3-22 shows the bit assignments.

Table 3-22 Overflow Flag Status Register

Bits	Reset value	Access	Function
[31:1]	-	RAZ/WI	Reserved
[0]	0x0	R/W	Event counter and cycle counter overflow flag

When reading this register, any overflow flag that reads as 0 indicates that the counter has not overflowed. An overflow flag that reads as 1 indicates that the counter has overflowed.

When writing to this register, any overflow flag written with a value of 0 is ignored, that is, no change. An overflow flag written with a value of 1 clears the counter overflow flag. The negated counter overflow bits are exported from the CCI-400 on the **nEVNTCNTOVERFLOW[4:0]** signal. You can use this to trigger interrupts. The MSB corresponds to the cycle count overflow.

3.4 Address map

The CCI-400 has a global address map, that is, every master has the same view of memory. This is split into 16 regions across the 40-bit address. The decode for each region is determined using a number of CCI-400 inputs that are expected to be static, that is, they are sampled only at reset. The inputs are **ADDRMAPx[1:0]**, where x is an integer from 0-15.

Figure 3-1 shows the CCI-400 address map.

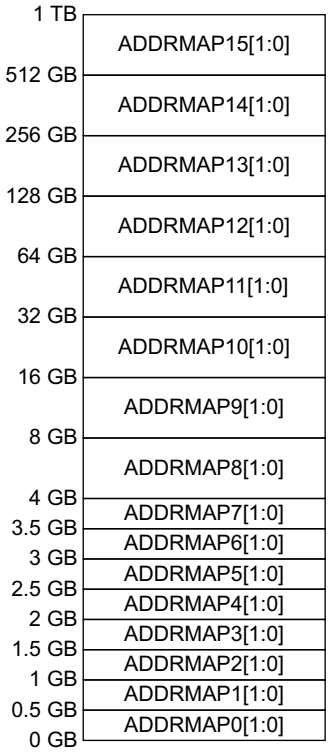

Figure 3-1 Address map

Table 3-23 shows the decoder mapping.

Table 3-23 Decoder mapping

ADDRMAPx[1:0]	Decode
0b00	M0, expected to be connected to the system
0b01	M1, expected to be connected to a memory controller
0b10	M2, expected to be connected to a memory controller
0b11	M1 and M2, striped in 4KB regions, used to load-balance between two memory controllers

The base address for internal CCI-400 registers is defined using a static input, **PERIPHBASE[39:15]**. The CCI-400 registers are offset by 0x90000 from this base address and occupy an address region of size 64KB.

For example, if **PERIPHBASE** is 0x0000000, then the register space occupies the following region:

0x0000090000 to 0x000009FFFF.

Accesses within this region, but not to a valid CCI-400 register, generate a DECERR response.

You can only access the CCI-400 register map through the system master interface, M0, so you must configure **PERIPHBASE** so that the **PERIPHBASE** to **PERIPHBASE** + `0x90000` region is within a single region that decodes to M0. An OVL assertion is included to test this assumption.

Note

ARM recommends that **PERIPHBASE[39:15]** occupies the bottom 4GB address space.

Appendix A
Signal Descriptions

This appendix describes the external signals of the CCI-400. This appendix contains the following section:

- *Signal descriptions* on page A-2.

A.1 Signal descriptions

This section describes the CCI-400 signals and contains the following subsections:

- *Clock and reset signals*
- *Configuration signals*
- *Debug signals* on page A-3
- *Slave interface signals* on page A-3
- *Clock and reset signals*.

A.1.1 Clock and reset signals

Table A-1 shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Direction	Description
ACLK	Input	Global clock
ARESETn	Input	Global reset

A.1.2 Configuration signals

Configuration signals are sampled only when **ARESETn** transitions from LOW to HIGH.

Table A-2 shows the configuration signals.

Table A-2 Configuration signals

Signal	Direction	Description
PERIPHBASE[39:15]	Input	Base address for CCI-400 programmable registers.[a]
ADDRMAPx[1:0][b]	Input	Defines the decode of each region of the address map. One set of inputs exists for each of the 16 regions in the address map.
BROADCASTCACHEMAINT[2:0]	Input	Broadcast cache maintenance operations. One bit exists for each master interface.
BARRIERTERMINATE[2:0]	Input	Terminate barriers, instead of propagating. One bit exists for each master interface.
BUFFERABLEOVERRIDE[2:0]	Input	Override the **AWCACHE[0]** and **ARCACHE[0]** outputs to be non-bufferable. One bit exists for each master interface.
QOSOVERRIDE[4:0]	Input	Override the **ARQOS** and **AWQOS** input signals. One bit exists for each slave interface.
ACCHANNELEN[4:0]	Input	If LOW, then AC requests are never issued on the corresponding slave interface. One bit exists for each slave interface.
ECOREVNUM[3:0]	Input	You must tie these signals LOW unless you have an *Engineering Change Order* (ECO) from ARM.

a. The base address for internal CCI-400 registers is defined using a static input, **PERIPHBASE[39:15]**. The CCI-400 registers are offset by `0x90000` from this base address, and occupy an address region of size 64KB. For example, if PERIPHBASE is `0x0000000`, then the register space occupies the region `0x0000090000` to `0x000009FFFF`.

b. Where x is 0-15.

A.1.3 Debug signals

The inputs can change at run-time and you must synchronize them to the CCI-400 clock to prevent timing hazards. Table A-3 shows the debug signals.

Table A-3 Debug signals

Signal	Direction	Description
NIDEN	Input	Non-invasive debug enable. If HIGH, enables the counting and export of PMU events.
SPNIDEN	Input	Secure privileged non-invasive debug enable. If HIGH, enables the counting of both non-secure and secure event.
EVNTBUS[120:0]	Output	CCI-400 events, exported if enabled in the PMCR. See *Event list* on page 2-5 for information on the pin allocations of this vector.
nEVNTCNTOVERFLOW[4:0]	Output	Overflow flags for the PMU clock and counters, active LOW. The CCNT overflow is bit 4, event counters on bits [3:0]
nERRORIRQ	Output	Indicates that an error response, DECERR or SLVERR, has been received on the **RRESP** or **BRESP** inputs, that cannot be signalled precisely. If LOW, indicates that an error has occurred.

A.1.4 DFT signal

Table A-4 shows the *Design For Test* (DFT) signal.

Table A-4 DFT signal

Signal	Direction	Description
DFTRSTDISABLE	Input	Disable reset during scan shift

A.1.5 Slave interface signals

A set of slave interface signals exists for each slave interface. The suffix is Sx, where x is 0-4.

This section describes:

- *Write address channel signals* on page A-4
- *Write data channel signals* on page A-4
- *Write data response channel signals* on page A-5
- *Read address channel signals* on page A-5
- *Read data channel signals* on page A-6
- *Coherency address channel signals* on page A-6
- *Coherency response channel signals* on page A-6
- *Coherency data channel signals, full ACE interfaces, S3 and S4 only* on page A-7
- *Acknowledge signals, full ACE interfaces, S3 and S4 only* on page A-7.

Write address channel signals

Table A-5 shows the write address channel signals.

Table A-5 Write address channel signals

Signal	Direction	Description
AWIDSx[n:0]	Input	Write address ID. You can configure the width of this signal.
AWADDRSx[39:0]	Input	Write address.
AWREGIONSx[3:0]	Input	Write address region. You can tie this signal LOW if the master does not drive it.
AWLENSx[7:0]	Input	Write burst length.
AWSIZESx[2:0]	Input	Write burst size.
AWBURSTSx[1:0]	Input	Write burst type
AWLOCKSx	Input	Write lock type.
AWCACHESx[3:0]	Input	Write cache type.
AWPROTSx[2:0]	Input	Write protection type.
AWSNOOPSx[2:0]	Input	Write snoop request type.
AWDOMAINSx[1:0]	Input	Write domain.
AWBARSx[1:0]	Input	Write barrier type.
AWQOSSx[3:0]	Input	Write *Quality-of-Service* (QoS) value.
AWUSERSx[n:0]	Input	User-specified extension to AW payload.
AWVALIDSx	Input	Write address valid.
AWREADYSx	Output	Write address ready.

Write data channel signals

Table A-6 shows the write data channel signals.

Table A-6 Write data channel signals

Signal	Direction	Description
WDATASx[127:0]	Input	Write data
WSTRBSx[15:0]	Input	Write byte-lane strobes
WLASTSx	Input	Write data last transfer indication
WUSERSx[n:0]	Input	User-specified extension to W payload
WVALIDSx	Input	Write data valid
WREADYSx	Output	Write data ready

Write data response channel signals

Table A-7 shows the write data response channel signals.

Table A-7 Write data response channel signals

Signal	Direction	Description
BIDSx[n:0]	Output	Write response ID. You can configure the width.
BRESPSx[1:0]	Output	Write response.
BUSERSx[n:0]	Output	User-specified extension to B payload.
BVALIDSx	Output	Write response valid.
BREADYSx	Input	Write response ready.

Read address channel signals

Table A-8 shows the read address channel signals.

Table A-8 Read address channel signals

Signal	Direction	Description
ARIDSx[n:0]	Input	Read address ID. You can configure the width of this signal.
ARADDRSx[39:0]	Input	Read address.
ARREGIONSx[3:0]	Input	Read address region. You can tie this signal LOW if the master does not drive it.
ARLENSx[7:0]	Input	Read burst length.
ARSIZESx[2:0]	Input	Read burst size.
ARBURSTSx[1:0]	Input	Read burst type.
ARLOCKSx	Input	Read lock type.
ARCACHESx[3:0]	Input	Read cache type.
ARPROTSx[2:0]	Input	Read protection type.
ARDOMAINSx[1:0]	Input	Read domain.
ARSNOOPSx[3:0]	Input	Read snoop request type.
ARBARSx[1:0]	Input	Read barriers.
ARQOSSx[3:0]	Input	Read QoS.
ARUSERSx[n:0]	Input	User-specified extension to AR payload.
ARVALIDSx	Input	Read address valid.
ARREADYSx	Output	Read address ready.

Read data channel signals

Table A-9 shows the read data channel signals.

Table A-9 Read data channel signals

Signal	Direction	Description
RIDSx[n:0]	Output	Read data ID. You can configure the width of this signal.
RDATASx[127:0]	Output	Read data.
RRESPSx[3:0]	Output	Read data response for ACE interfaces, that is, S3 and S4.
RRESPSx[1:0]	Output	Read data response for ACE-Lite interfaces, that is, S0, S1, S2.
RLASTSx	Output	Read data last transfer indication.
RUSERSx[n:0]	Output	User-specified extension to R payload.
RVALIDSx	Output	Read data valid.
RREADYSx	Input	Read data ready.

Coherency address channel signals

Table A-10 shows the coherency address channel signals.

Table A-10 Coherency address channel signals

Signal	Direction	Description
ACADDRSx[39:0]	Output	Snoop address
ACPROTSx[2:0]	Output	Snoop protection type
ACSNOOPSx[3:0]	Output	Snoop request type
ACVALIDSx	Output	Snoop address valid
ACREADYSx	Input	Master ready to accept snoop address

Coherency response channel signals

Table A-11 shows the coherency response channel signals.

Table A-11 Coherency response channel signals

Signal	Direction	Description
CRRESPSx[4:0]	Input	Snoop response
CRVALIDSx	Input	Snoop response valid
CRREADYSx	Output	Slave ready to accept snoop response

Coherency data channel signals, full ACE interfaces, S3 and S4 only

Table A-12 shows the coherency data channel signals, for full ACE interfaces, S3 and S4 only.

Table A-12 Coherency data channel signals, full ACE interfaces, S3 and S4 only

Signal	Direction	Description
CDDATASx[127:0]	Input	Snoop data
CDLASTSx	Input	Snoop data last transfer
CDVALIDSx	Input	Snoop data valid
CDREADYSx	Output	Slave ready to accept snoop data

Acknowledge signals, full ACE interfaces, S3 and S4 only

Table A-13 shows the acknowledge signals, full ACE interfaces, S3 and S4 only.

Table A-13 Acknowledge signals, full ACE interfaces, S3 and S4 only

Signal	Direction	Description
RACKSx	Input	Read acknowledge
WACKSx	Input	Write acknowledge

A.1.6 Master interface signals

A set of master interface signals exists for each master interface. The suffix is My, where y is 0, 1, or 2.

This section describes:

- *Write address channel signals*
- *Write data channel signals* on page A-8
- *Write data response channel signals* on page A-8
- *Read address channel signals* on page A-9
- *Read data channel signals* on page A-9
- *Power control signals, C-channel* on page A-10.

Write address channel signals

Table A-14 shows the write address channel signals.

Table A-14 Write address channel signals

Signal	Direction	Description
AWIDMy[n:0]	Output	Write address ID. Width is the maximum **AWID** width across the slave interfaces + 3 bits.
AWADDRMy[39:0]	Output	Write address.
AWREGIONMy[3:0]	Output	Write address region.
AWLENMy[7:0]	Output	Write burst length.
AWSIZEMy[2:0]	Output	Write burst size.

Table A-14 Write address channel signals (continued)

Signal	Direction	Description
AWBURSTMy[1:0]	Output	Write burst type.
AWLOCKMy	Output	Write lock type.
AWCACHEMy[3:0]	Output	Write cache type.
AWPROTMy[2:0]	Output	Write protection type.
AWSNOOPMy[2:0]	Output	Write snoop request type.
AWDOMAINMy[1:0]	Output	Write domain.
AWBARMy[1:0]	Output	Write barrier type.
AWQOSMy[3:0]	Output	Write QoS value.
AWUSERMy[n:0]	Output	User-specified extension to AW payload.
AWVALIDMy	Output	Write address valid.
AWREADYMy	Input	Write address ready.

Write data channel signals

Table A-15 shows the write data channel signals.

Table A-15 Write data channel signals

Signal	Direction	Description
WDATAMy[127:0]	Output	Write data
WSTRBMy[15:0]	Output	Write byte-lane strobes
WLASTMy	Output	Write data last transfer indication
WUSERMy[n:0]	Output	User-specified extension to W payload
WVALIDMy	Output	Write data valid
WREADYMy	Input	Write data ready

Write data response channel signals

Table A-16 shows the write data response channel signals.

Table A-16 Write data response channel signals

Signal	Direction	Description
BIDMy[n:0]	Input	Write response ID
BRESPMy[1:0]	Input	Write response
BUSERMy[n:0]	Input	User-specified extension to B payload
BVALIDMy	Input	Write response valid
BREADYMy	Output	Write response ready

Read address channel signals

Table A-17 shows the read address channel signals.

Table A-17 Read address channel signals

Signal	Direction	Description
ARIDMy[n:0]	Output	Read address ID. Width is the maximum **ARID** width across slave interfaces + 3 bits.
ARADDRMy[39:0]	Output	Read address.
ARREGIONMy[3:0]	Output	Read address region.
ARLENMy[7:0]	Output	Read burst length.
ARSIZEMy[2:0]	Output	Read burst size.
ARBURSTMy[1:0]	Output	Read burst type.
ARLOCKMy	Output	Read lock type.
ARCACHEMy[3:0]	Output	Read cache type.
ARPROTMy[2:0]	Output	Read protection type.
ARDOMAINMy[1:0]	Output	Read domain.
ARSNOOPMy[3:0]	Output	Read snoop request type.
ARBARMy[1:0]	Output	Read barriers.
ARQOSMy[3:0]	Output	Read QoS value.
ARUSERMy[n:0]	Output	User-specified extension to AR payload.
ARVALIDMy	Output	Read address valid.
ARREADYMy	Input	Read address ready.

Read data channel signals

Table A-18 shows the read data channel signals.

Table A-18 Read data channel signals

Signal	Direction	Description
RIDMy[n:0]	Input	Read data ID
RDATAMy[127:0]	Input	Read data
RRESPMy[1:0]	Input	Read data response
RLASTMy	Input	Read data last transfer indication
RUSERMy[n:0]	Input	User-specified extension to R payload
RVALIDMy	Input	Read data valid
RREADYMy	Output	Read data ready

Power control signals, C-channel

Table A-19 shows the power control signals, C-channel.

Table A-19 Power control signals, C-channel

Signal	Direction	Description
ACTIVEMy	Output	Indicates that the master interface has active transactions. You can use it to gate the clock to downstream components.
CSYSREQ	Input	Request to disable the **ACLK** input.
CSYSACK	Output	Clock disable response.
CACTIVE	Output	Indicates that the CCI-400 requires the **ACLK** input to run.

For information on using the power control signals, see the *CoreLink CCI-400 Cache Coherent Interconnect Integration Manual*.

Appendix B
Revisions

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

Table B-1 Issue A

Change	Location	Affects
First release	-	-

Table B-2 Differences between issue A and issue B

Change	Location	Affects
Added a new section on exclusive accesses	*Exclusive accesses* on page 2-15	All revisions
Removed the erroneous reference to the TSPEC regulator	*Regulation based on outstanding transactions* on page 2-18	All revisions
Changed the Component and Peripheral ID registers table so that the cells are in order of address offset instead of register name	Table 3-9 on page 3-11	All revisions
Updated the revision number in the Peripheral ID2 register	Table 3-9 on page 3-11	r0p1

Table B-3 Differences between issue B and issue C

Change	Location	Affects
Updated the parameter values for maximum ID width on slave interfaces and maximum size of read and write trackers in master interfaces	*Product revisions* on page 1-10	r0p2
Changed the configuration of read and write trackers in master interfaces	*Product revisions* on page 1-10	r0p2
Added a description for product improvements	*Product revisions* on page 1-10	r0p2
Added a note to the Event list section	*Event list* on page 2-5	All revisions
Updated the revision number in the Peripheral ID2 register	Table 3-9 on page 3-11	r0p2
Added more information to the description of the Shareable Override Register	*Shareable Override Register* on page 3-13	All revisions
Added a description of how to calculate the number of outstanding transactions for the MAX OT Registers	*Max OT Registers* on page 3-15	All revisions
Changed the bit range of the **RRESPSx** signal to cover ACE interfaces and ACE-Lite interfaces. For ACE interfaces it is **RRESPSx[3:0]** and for ACE-Lite interfaces it is **RRESPSx[1:0]**.	Table A-9 on page A-6	All revisions
Changed the bit range of the **RRESPMy** signal from [3:0] to [1:0] so it becomes **RRESPMy[1:0]** instead of **RRESPMy[3:0]**	Table A-18 on page A-9	All revisions

Table B-4 Differences between issue C and issue D

Change	Location	Affects
Note added to *QoS value based on latency*	*QoS value based on latency measurement* on page 2-17	All revisions
Regulation based on outstanding transactions moved to new section	*Regulation based on outstanding transactions* on page 2-18	All revisions
Repeat the explanation about the PERIPHBASE offset at strategic locations	*Register summary* on page 3-3 and Table A-2 on page A-2	All revisions
Configurations within Target Latency Registers updated	*Target Latency Registers* on page 3-16	All revisions
Configurations within Latency Regulation Registers updated	*Latency Regulation Registers* on page 3-16	All revisions
Configurations within QoS Range Register updated	*QoS Range Register* on page 3-17	All revisions
ACBARSx signal removed from table because it is not used	Table A-10 on page A-6	All revisions