CoreLink™ NIC-400 Network Interconnect

Revision: r0p0

Technical Reference Manual

ARM®

ARM DDI 0475A (ID081712)

CoreLink NIC-400 Network Interconnect
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
07 August 2012	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
CoreLink NIC-400 Network Interconnect Technical Reference Manual

Preface
About this book vi
Feedback ix

Chapter 1 Introduction
1.1 About the CoreLink NIC-400 Network Interconnect 1-2
1.2 Key features 1-3
1.3 Relationship between NIC-400 and AMBA Designer 1-5
1.4 Product revisions 1-7

Chapter 2 Functional Description
2.1 About the functions 2-2
2.2 Interfaces 2-3
2.3 Operation 2-15
2.4 Optional features 2-29

Chapter 3 Programmers Model
3.1 About the programmers model 3-2
3.2 Configuration programmers model 3-3

Appendix A Signal Descriptions
A.1 Global signals A-2
A.2 Signal direction A-3
A.3 AXI3 and AXI4 signals A-4
A.4 APB signals A-9
A.5 AHB-Lite signals A-10

A.6 QVN signals A-14

Appendix B Revisions

Preface

This preface introduces the *CoreLink NIC-400 Network Interconnect Technical Reference Manual*. It contains the following sections:

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

About this book

This book is for the CoreLink NIC-400 Network Interconnect (NIC-400).

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 AMBA Network Interconnect.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to NIC-400 and a description of its features.

Chapter 2 *Functional Description*

Read this for a description of the functionality of the NIC-400.

Chapter 3 *Programmers Model*

Read this for a description of the address map and registers of the NIC-400.

Appendix A *Signal Descriptions*

Read this for a description of the signals used by the NIC-400.

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*
- *Signals* on page vii.

Typographical

The typographical conventions are:

italic Introduces special terminology, denotes cross-references, and citations.

bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`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 QoS-400 Network Interconnect Advanced Quality of Service, Supplement to CoreLink NIC-400 Network Interconnect Technical Reference Manual* (ARM DSU 0026).

  Note

  This product is separately licensed and not included in the NIC-400 base product.

- *CoreLink QVN-400 Network Interconnect Advanced Quality of Service for Virtual Networks, Supplement to CoreLink NIC-400 Network Interconnect Technical Reference Manual* (ARM DSU 0027).

  Note

  This product is separately licensed and not included in the NIC-400 base product.

- *CoreLink TLX-400 Network Interconnect Thin Links, Supplement to CoreLink NIC-400 Network Interconnect Technical Reference Manual* (ARM DSU 028).

  Note

  This product is separately licensed and not included in the NIC-400 base product.

- *AMBA Designer ADR-400 User Guide* (ARM DUI 0333).
- *AMBA 4 AXI4, AXI4-Lite, and AXI4-Stream Protocol Assertions User Guide* (ARM DUI 0534).
- *AMBA Specification* (ARM IHI 0011).
- *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022).
- *AMBA APB Protocol Specification* (ARM IHI 0024).
- *AMBA 4 AXI4-Stream Protocol Specification* (ARM IHI 0051).

The following confidential books are only available to licensees:

- *CoreLink NIC-400 Network Interconnect Integration Manual* (ARM DII 0269).
- *CoreLink NIC-400 Network Interconnect Implementation Guide* (ARM DII 0273).
- *CoreLink NIC-400 Network Interconnect Supplement to CoreLink ADR-400 AMBA Designer User Guide* (ARM DSU 0018).
- *CoreLink QVN Protocol Specification* (ARM IHI 0063).

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 0475A
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the CoreLink™ NIC-400 Network Interconnect. It contains the following sections:

- *About the CoreLink NIC-400 Network Interconnect* on page 1-2
- *Key features* on page 1-3
- *Relationship between NIC-400 and AMBA Designer* on page 1-5
- *Product revisions* on page 1-7.

1.1 About the CoreLink NIC-400 Network Interconnect

The CoreLink NIC-400 Network Interconnect is highly configurable and enables you to create a complete high performance, optimized AMBA-compliant network infrastructure. The possible configurations for the CoreLink NIC-400 Network Interconnect can range from a single bridge component, for example an AHB to AXI protocol conversion bridge, to a complex infrastructure that consists of up to 128 masters and 64 slaves of AMBA protocols.

The NIC-400 configuration can consist of multiple switches with many topology options. Figure 1-1 shows a top-level block diagram of the NIC-400 that contains:

- multiple switches
- multiple *AMBA Slave Interface Blocks* (ASIBs)
- multiple *AMBA Master Interface Blocks* (AMIBs).

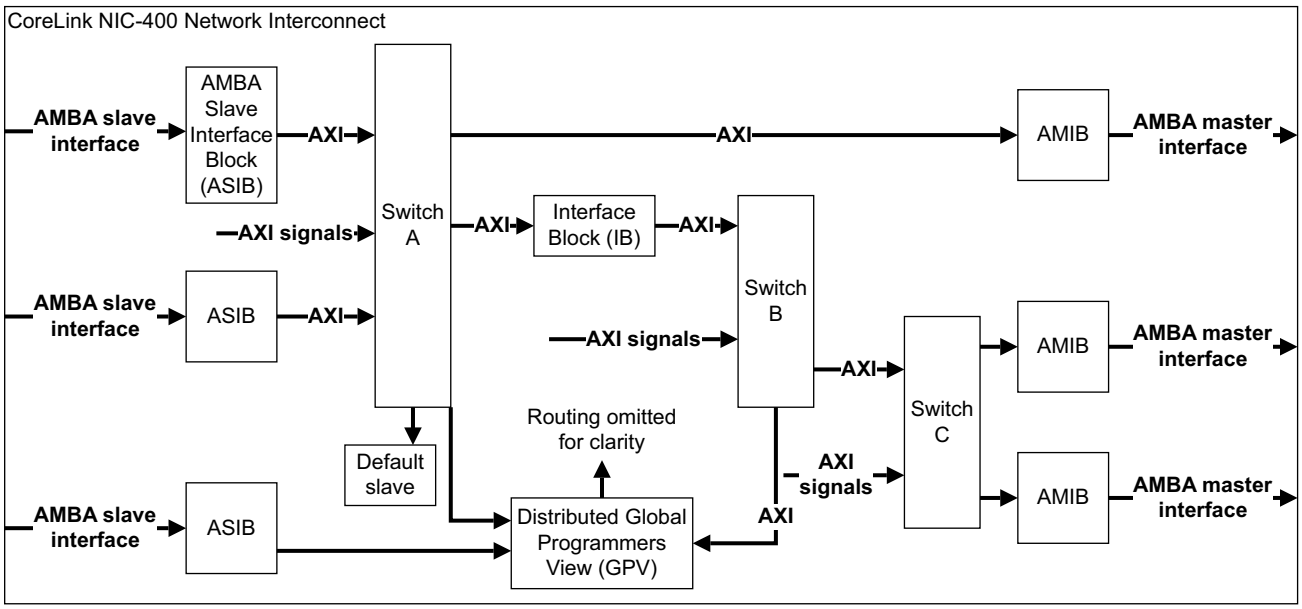

Figure 1-1 CoreLink NIC-400 Network Interconnect top-level block diagram

1.2 Key features

The CoreLink NIC-400 Network Interconnect is a highly configurable infrastructure component that supports:

- 1-128 slave interfaces that can be:
  - AXI3
  - AXI4
  - AHB-Lite Slave
  - AHB-Lite Mirrored Master.
- 1-64 master interfaces that can be:
  - AXI3
  - AXI4
  - AHB-Lite Master
  - AHB-Lite Mirrored Slave
  - APB2
  - APB3
  - APB4.
- Hierarchical clock gating.
- Configuration of:
  - an APB port to support 1-16 slaves

    Note

    an APB master interface that can have up to 16 ports of APB2, APB3 and APB4
  - an AXI port to support four region control bits
  - an AXI port to support QoS signalling.
- Single-cycle arbitration.
- Full pipelining to prevent master stalls.
- Programmable control for FIFO transaction release.
- Multiple switch networks.
- Complex topologies, including *Network on Chip* (NoC) loop-back connections between switches.
- Up to five cascaded switch networks between any master and slave interface pair.
- AXI or AHB-Lite masters and slaves with:
  - an address width of 32-64 bits
  - a data width of 32, 64, 128, or 256 bits.
- Non-contiguous APB slave address map for a single master interface.
- Independent widths of user-defined sideband signals for each channel.
- *Global Programmers View* (GPV) for the entire infrastructure that you can configure so any master, or a discrete configuration slave interface, can access it. See Chapter 3 *Programmers Model*.
- Highly flexible timing closure options.
- Hierarchical clock gating to reduce idle or near idle power.

- *Quality of Service* (QoS), using the QoS-400 product. See also *Optional features* on page 2-29.

  Note

  This product is separately licensed and not included in the NIC-400 base product.

- *QoS Virtual Networks* (QVN), using the QVN-400 product. See also *Optional features* on page 2-29.

  Note

  This product is separately licensed and not included in the NIC-400 base product.

- *Thin Links* (TLX), using the TLX-400 product. See also *Optional features* on page 2-29.

  Note

  This product is separately licensed and not included in the NIC-400 base product.

1.3 Relationship between NIC-400 and AMBA Designer

AMBA Designer is a configuration tool that generates a specific implementation of a CoreLink NIC-400 Network Interconnect. AMBA Designer drives the CoreLink NIC-400 Network Interconnect generation engine to provide the following for a set of configuration parameters and implementation scripts:

- Verilog *Register Transfer Level* (RTL)
- testbench and stimulus
- synthesis scripts.

The documentation suites and implementation scripts for the CoreLink NIC-400 Network Interconnect and AMBA Designer are designed to be used together to describe the principles of the CoreLink NIC-400 Network Interconnect and the actual configuration options. There is no duplication between the two sets of documentation. The following sections describe the information that each documentation suite provides:

- *CoreLink NIC-400 Network Interconnect documentation*
- *AMBA Designer documentation*.

1.3.1 CoreLink NIC-400 Network Interconnect documentation

The CoreLink NIC-400 Network Interconnect documentation consists of:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes how to create the transfer function and possible capabilities of the network component and how to dynamically change it using the programmers model.

Implementation Guide

The *Implementation Guide* (IG) describes how to set up the network environment and how to use it to run RTL simulations or implementation scripts.

Integration Manual

The *Integration Manual* (IM) describes how to integrate a configured network into a larger subsystem.

1.3.2 AMBA Designer documentation

The AMBA Designer User Guide describes how to:

- Install AMBA Designer.
- Generate and verify RTL sub-systems of ARM IP.
- Stitch ARM components together. ARM components conform to the IP-XACT™ standard from the SPIRIT Consortium™.

The CoreLink Supplement to AMBA Designer User Guide describes how to produce a customized infrastructure.

Note

The supplement provides configuration guidance.

1.3.3 Documentation for optional CoreLink features

The optional CoreLink NIC-400 Network Interconnect documentation consists of:

QoS Supplement to TRM

The QoS Supplement to TRM describes programmable QoS facilities for attached AMBA masters that support read and write QoS requests.

QVN Supplement to TRM

The QVN Supplement to TRM describes a mechanism to avoid head of line blocking and cross path blocking between different data flows.

TLX Supplement to TRM

The TLX Supplement to TRM describes a mechanism to reduce the number of signals in an AXI point-to-point connection and enable it to be routed over a longer distance.

Note

These optional features are obtained through additional licences for QoS-400, QVN-400 and TLX-400 to that required for the NIC-400 base product.

1.4 Product revisions

This section describes the differences in functionality between product revisions of the CoreLink NIC-400 Network Interconnect:

r0p0 First release.

Chapter 2
Functional Description

This chapter describes the functionality of the CoreLink™ NIC-400 Network Interconnect. It contains the following sections:

- *About the functions* on page 2-2
- *Interfaces* on page 2-3
- *Operation* on page 2-15
- *Optional features* on page 2-29.

2.1 About the functions

The CoreLink NIC-400 Network Interconnect is built from functions, each with its own transfer function. A transfer function can be:

- a domain crossing, for example:
  - clock domain crossing
  - data width crossing.
- used to create timing isolation, for optimizing critical network paths for latency.

Within a domain, a switch, or multiple switches, can exist to enable routing paths between any slave interface to any master interface.

The functions are configured into routing switches or *Interface Blocks* (IBs) and you can use AMBA Designer to create highly complex topologies using these modules.

2.2 Interfaces

This section describes the CoreLink NIC-400 Network Interconnect interfaces and contains the following subsections:

- *Slave interfaces*
- *Master interfaces* on page 2-8
- *Low-power interfaces, clock-gating* on page 2-13.

2.2.1 Slave interfaces

The CoreLink NIC-400 Network Interconnect supports the following slave interfaces:

- *AXI3 and AXI4 slave interfaces*
- *AHB-Lite slave interfaces* on page 2-4.

Note

Any transaction that does not decode to a legal master interface destination, or programmers view register, receives a DECERR response. For an AHB master, the AXI DECERR is mapped to an AHB ERROR.

The AXI DECERR error is mapped to an AHB master ERROR if:

- you do not configure the early write response
- you configure INCR Promotion and Early Write Response and the transaction is non-cacheable
- the AHB burst is not broken.

AXI3 and AXI4 slave interfaces

An AXI slave interface supports the AXI protocol.

Note

- NIC-400 base product does not accept, or issue interleaved write data.
- Data widths of 512 or 1024 bits are not supported.

Configuration options

You can configure the following options:

- Address width of 32-64 bits.
- Data width of 32, 64, 128, or 256 bits.
- User sideband signal width of 0-256 bits.
- Data width upsizer function. See *Upsizing data width function* on page 2-16.
- Data width downsizer function. See *Downsizing data width function* on page 2-18.
- Frequency domain crossing of the following types:
  - ASYNC
  - SYNC 1:1
  - SYNC 1:n
  - SYNC n:1

  - SYNC n:m.

- Security of the following types:

  Secure All transactions originating from this slave interface are flagged as secure transactions and can access both secure and non-secure components.

  Non-secure

  All transactions originating from this slave interface are flagged as non-secure transactions and cannot access secure components.

  Per access

  The **AxPROT[1]** signal determines the security setting of each transaction, and the slaves that it can access.

- Support for the full AXI protocol.

  Note

  - Data widths of 512 or 1024 bits are not supported.
  - You can achieve a gate count reduction and a performance increase if the attached master does not create any AXI3 lock transactions.

- Write acceptance capability of 1-32 transactions.

  Note

  If buffering components exist within the ASIB, then this value can be higher. For example, a full register slice in the slave interface position of the ASIB increases the write acceptance capability by two, and a forward register slice in the same position increases the write acceptance capability by one.

- Read acceptance capability of 1-127 transactions.

  Note

  If buffering components exist within the ASIB, then this value can be higher. For example, a full register slice in the slave interface position of the ASIB increases the read acceptance capability by two, and a forward register slice in the same position increases the read acceptance capability by one.

- Buffering, see *FIFO and clocking function* on page 2-20.
- Timing isolation:
  - from the external master
  - from the infrastructure.

AHB-Lite slave interfaces

The CoreLink NIC-400 Network Interconnect can support the full AHB-Lite protocol using either:

- an AHB-Lite slave interface
- an AHB-Lite mirror master interface.

The following configuration options can improve AHB-Lite to AXI performance, but cannot always be used robustly:

- **INCR promotion and Early Write Response**
- **allow broken bursts**.

If you configure the interface as an AHB mirror master interface, you cannot configure **allow broken bursts** because the AHB-Lite protocol does not permit AHB-Lite masters to break bursts.

Table 2-1 shows the four combinations for the configuration of **INCR promotion and Early Write Response** and **allow broken bursts** and contains links to descriptions for each option.

Table 2-1 Combination of configuration parameters

INCR promotion and Early Write Response	allow broken bursts	Description of combination
Configured	Not configured	*Combination 1*
Not configured	Configured	*Combination 2*
Configured	Configured	*Combination 3* on page 2-6
Not configured	Not configured	*Combination 4* on page 2-6

Combination 1

If you configure **INCR promotion and Early Write Response** and do not configure **allow broken bursts** then the network converts all:

- AHB read fixed length bursts to AXI fixed length bursts.
- AHB write fixed length bursts with **HPROT[3]** asserted to AXI fixed length bursts:
  - All AHB write data beats receive an automatic OKAY response from the bridge irrespective of the B-channel AXI response. This means that if the network receives an error response, it does not feed it back to the master.
  - The bridge can support up to five outstanding write accesses.
- Write fixed-length bursts with **HPROT[3]** negated to AXI fixed length bursts, and only the last AHB write data beat receives the AXI buffered response for the complete AHB transaction.
- AHB read INCR bursts with **HPROT[3]** asserted to AXI INCR4 bursts.
- Write INCR bursts with **HPROT[3]** asserted to AXI INCR4 bursts, and all AHB write data beats receive an automatic OKAY response from the bridge, irrespective of the B-channel AXI response. This means that if the network receives an error response, it does not feed it back to the master.
- Read INCR bursts with **HPROT[3]** negated to a series of AXI singles.
- Write INCR bursts with **HPROT[3]** negated to a series of AXI singles, and each AHB write beat is acknowledged with the AXI buffered write response.

Combination 2

If you configure **allow broken bursts** and do not configure **INCR promotion and Early Write Response**, the network converts all:

- Read fixed length bursts with **HPROT[3]** asserted to AXI fixed length bursts.
- Read fixed length bursts with **HPROT[3]** negated to AXI singles.
- Write fixed length bursts with **HPROT[3]** asserted to AXI fixed length bursts, but only the last AHB write data beat receives the AXI buffered response for the whole AHB transaction. However, if the AHB burst is broken, then the network does not feed the AXI response back to the master.

- Write fixed length bursts with **HPROT[3]** negated to AXI singles, and each AHB write beat is acknowledged with the AXI buffered write response.
- Read INCR bursts to a series of AXI singles.
- Write INCR bursts to a series of AXI singles, and each AHB write beat is acknowledged with the AXI buffered write response.

Combination 3

If you configure **INCR promotion and Early Write Response** and configure **allow broken bursts** then the network converts all:

- Read fixed length bursts with **HPROT[3]** asserted to AXI fixed length bursts.
- Read fixed length bursts with **HPROT[3]** negated to AXI singles.
- Write fixed length bursts with **HPROT[3]** asserted to AXI fixed length bursts:
  - The bridge sends an automatic OKAY response to all the AHB write data beats, disregarding the B-channel AXI response. Therefore, if the network generates an error response, it does not feed it back to the master.
  - The bridge can support up to five outstanding write accesses because the RAW hazard detection function supports up to four transactions. A fifth write is issued, but the AHB write response is not issued until a slot is freed in the RAW hazard monitor.
- Write fixed length bursts with **HPROT[3]** negated to AXI singles, and each AHB write beat is acknowledged with the AXI buffered write response.
- Read INCR bursts with **HPROT[3]** asserted speculatively to AXI INCR4 bursts.
- Write INCR bursts with **HPROT[3]** asserted speculatively to AXI INCR4 bursts, and all AHB write data beat receive an automatic OKAY response from the bridge irrespective of the B-channel AXI response. Therefore, if the network generates an error response, it does not feed it back to the master.
- Read INCR bursts with **HPROT[3]** negated to a series of AXI singles.
- Write INCR bursts with **HPROT[3]** negated to a series of AXI singles, and each AHB write beat is acknowledged with the AXI buffered write response.

Combination 4

If you do not configure **INCR promotion and Early Write Response** and do not configure **allow broken bursts** then the network converts all:

- read fixed length bursts to AXI fixed length bursts
- write fixed length bursts to AXI fixed length bursts, and only the last AHB write data beat receives the AXI buffered response for the whole AHB transaction
- read INCR bursts to a series of AXI singles
- write INCR bursts to a series of AXI singles, and each AHB write beat is acknowledged with the AXI buffered write response.

Note

If you select either the **INCR promotion and Early Write Response** or **allow broken bursts** configuration options, or both, then the following programmable function override bits also exist:

`rd_incr_override` Converts all AHB read transactions to a series of AXI singles.

`wr_incr_override` Converts all AHB write transactions to a series of AXI singles.

You can configure these bits through a GPV port.

See Chapter 3 *Programmers Model* for more information.

Error response

If the AHB master cancels a burst when it receives an ERROR response, the bridge stalls the master until the network receives all the read data beats from the AXI domain. This is only possible with read transfers because AXI writes receive a response at the end of the burst only.

Note

When communicating with transfer-sensitive slave devices such as FIFOs, the master might not be aware of how many read data beats have been read.

Lock transactions

The only supported lock transactions are SWP locks. That is, a single locking read followed by a single unlocking write, with an undefined number of IDLE transactions in between.

Note

If the network receives a non-SWP lock sequence, it is possible for a network path to be stalled, particularly if an odd number of lock transactions is issued. The stall is cancelled on the next transaction received that unlocks the stalled path.

If you configure lock support and a GPV, then a lock override function is also configured. You can program this option, named `lock_override`, to force no AXI lock transactions to be created. See Chapter 3 *Programmers Model*.

Configuration options

You can configure the following AHB options:

- AHB slave or mirrored master interface types.
- Address width of 32-64 bits.
- Data width of 32, 64, 128, or 256 bits.
- Data width upsizer function that *Upsizing data width function* on page 2-16 describes.
- Data width downsizer function that *Downsizing data width function* on page 2-18 describes.
- Frequency domain crossing of the following types:
  - ASYNC
  - SYNC 1:1
  - SYNC 1:n

  - SYNC n:1
  - SYNC n:m.

- Security of the following types:

  Secure All transactions originating from this slave interface are flagged as secure transactions and can access both secure and non-secure components.

  Non-secure

  All transactions originating from this slave interface are flagged as non-secure transactions and cannot access secure components.

- INCR promotion and Early Write Response.
- Permit broken bursts using the `allow broken bursts` parameter.
- Support for the full AHB-Lite protocol with only SWP locks.

  Note

  You can reduce the gate count and increase the performance if the attached master does not create any AHB lock transactions.

- Timing isolation:
  - from the external master
  - from the network
  - user signals.

  Note

  HAUSER maps onto **AWUSER** or **ARUSER** internally depending on the access type.

  HWUSER maps onto **WUSER**.

  RUSER maps to **HRUSER**.

2.2.2 Master interfaces

The CoreLink NIC-400 Network Interconnect supports the following master interfaces:

- *AXI3 and AXI4 master interfaces*
- *AHB master interfaces* on page 2-10
- *APB master interfaces* on page 2-12.

AXI3 and AXI4 master interfaces

The network supports the AXI protocol using an AXI master interface.

Note

Data widths of 512 or 1024 bits are not supported.

Configuration options

You can configure the following AXI options:

- Address width of 32-64 bits.
- Data width of 32, 64, 128, or 256 bits.
- Data width upsizer function that *Upsizing data width function* on page 2-16 describes.

- User sideband signal width of 0-256 bits.
- Data width downsizer function that *Downsizing data width function* on page 2-18 describes.
- Frequency domain crossing of type:
  - ASYNC
  - SYNC 1:1
  - SYNC 1:n
  - SYNC n:1
  - SYNC n:m.
- Support for the full AXI protocol.

  Note

  - You can reduce the gate count and increase the performance if all attached masters that can access the master interface do not create any AXI lock transactions.
  - Data widths of 512 or 1024 bits are not supported.
- Write issuing capability of 1-32 transactions.
- Read issuing capability of 1-127 transactions.

  Note

  You can configure the read issuing capability as 0 when a master interface is not also configured as upsizing or downsizing. The value of 0 removes any limiting of the read issuing capability by the interface. The read issuing capability is the sum of all upstream nodes that can access the interface.
- Buffering that *FIFO and clocking function* on page 2-20 describes.
- Timing isolation:
  - from the external slave
  - from the network.
- AXI masters, you can reduce the number of ID bits exported at the master interface, see *Global ID and ID reduction* on page 2-28.
- AXI region:
  - You can determine an AXI region value for a slave by applying an additional finer granularity at the address decode. Alternatively, you can input a region from the master interface. The AXI region is output to all slaves that have **Multi-region Slave** selected.

    Note

    You can select a 4-bit output region for a slave value, or you can input a region from the master interface.
  - If an APB slave is addressed, then an input region is overridden by the full address decode.

AHB master interfaces

The network can support the full AHB-Lite master protocol and you can configure the network to provide an AHB-Lite mirrored slave protocol. Table 2-2 shows the mapping of AXI burst types to AHB burst types.

Table 2-2 AXI burst type to AHB burst type mapping

AxBURST	Number of transfers in AXI transaction	HBURST	Notes
FIXED	-	SINGLE	This is a series of singles, and the number depends on the AxLEN setting
INCR	1	SINGLE	-
-	4	INCR4	-
-	8	INCR8	-
-	16	INCR16	-
-	2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15 AXI4 extends burst length support for the INCR burst type to 2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17-256	INCR	Undefined length
WRAP	2	SINGLE	Two transfers
WRAP	4	WRAP4	-
-	8	WRAP8	-
-	16	WRAP16	-

Note

Transactions from AHB slave interfaces that are configured with early write response or broken bursts are output as INCR transactions of an undefined length.

If the AHB protocol conversion function receives an unaligned address, or a write data beat without all the byte strobes set, the CoreLink NIC-400 Network Interconnect detects it, and a programmable enable bit permits the network to create a DECERR response.

Note

- Because AHB-Lite does not support **WSTRBs**, when accessing AHB slaves from an AXI master, care must be taken not to generate transactions have partial strobes. An example of this would be when a AXI master is accessing a AHB slave, instead of issuing a single 32-bit transaction with **WSTRB** `b0110` you must issue two 8-bit transactions.
- If you set the `force_incr` programmable bit, see Table 3-3 on page 3-7, and a beat is received that has no write data strobes set, that write data beat is replaced with an IDLE beat.
- You can configure the inclusion of the programmable enable bit to create a reduced gate count implementation.

See Chapter 3 *Programmers Model*. The network still transmits the unaligned address transfer into the AHB domain, but it aligns the address by forcing the lower address bits of the size of the transaction to zeros.

The network breaks any transactions that cross a 1KB boundary into multiple AHB INCR bursts. You can configure a programmable option, named `force_incr`, see Table 3-3 on page 3-7, that maps all transactions that are to be output to the AHB domain to be an undefined length INCR.

If the AXI burst is part of a locked sequence, the AHB-Lite translation keeps **HMASTLOCK** asserted across the boundary to ensure that the burst atomicity is not compromised. For write transactions, AHB responses are merged into a single AXI buffered response. The merged response is an AXI SLAVE ERROR if any of the AHB-Lite data beats have an AHB ERROR.

Any transaction that the network receives without all WSTRBs asserted or negated still goes ahead. This means that erroneous data bytes might be written to the slave.

Configuration options

You can configure the following options for the AHB interface:

- AHB master or mirrored slave interface types.
- Address width of 32-64 bits.
- Data width of 32, 64, 128, or 256 bits.
- Data width upsizer function that *Upsizing data width function* on page 2-16 describes.
- Data width downsizer function that *Downsizing data width function* on page 2-18 describes.
- Frequency domain crossing of the following types:
  - ASYNC
  - SYNC 1:1
  - SYNC 1:n
  - SYNC n:1
  - SYNC n:m.
- Security of the following types:

  Secure Only secure transactions can access components attached to this master interface.

  Non-secure

  Both secure and non-secure transactions can access components attached to this master interface.

  Boot time secure

  You can use software to configure whether it permits secure and non-secure transactions to access components attached to this master using the Secure and Non-secure options.
- Support for the full AHB-Lite master protocol.
- Timing isolation:
  - from the external slave
  - from the network
  - user signals.

  Note

  HAUSER maps onto **AWUSER** or **ARUSER** internally depending on the access type.

  HWUSER maps onto **WUSER**.

RUSER maps to **HRUSER**.

APB master interfaces

You can configure the APB interface to support a mixture of APB2, APB3, or APB4. The APB data width is always 32-bit, and it is therefore never necessary for the APB interface to require the upsizer function. The APB interface can ignore AXI writes strobes. If the network receives a write transaction with all of the write strobes negated, then it does not perform the write.

Note

APB SLVERR responses are converted to AXI SLVERR responses.

Any transaction that the network receives without all four WSTRBs asserted or negated still proceeds. This means that erroneous data bytes might be written to the APB3 or the ABP2 slaves.

Note

Because APB4 supports write strobes, the APB4 slave is unaffected by sparse data bytes.

The masters accessing the APB interface must ensure that only Word writes access the APB sub-system. The address and data widths are fixed as follows:

- address width of 32-bit
- data width of 32-bit.

Note

Although the CoreLink NIC-400 Network Interconnect only outputs 32 address bits, you can configure the APB address of any peripheral to be anywhere in the address map.

Configuration options

You can configure the following options:

- data width downsizer function as *Downsizing data width function* on page 2-18 describes
- frequency domain crossing for the majority of APB ports of the following types:
  - ASYNC
  - SYNC 1:1
  - SYNC 1:n
  - SYNC n:1
  - SYNC n:m.
- buffering as *FIFO and clocking function* on page 2-20 describes
- 1-16 supported APB slaves
- configurable address region sizes
- non-contiguous address regions
- you can configure each APB slave for:
  - APB2, APB3, or APB4
  - asynchronous interface to the majority of APB ports.

- security of the following types:
  - secure for each APB port
  - non-secure for each APB port
  - boot secure for all APB ports.

Note

- To configure an APB port as secure or non-secure, the parent AMIB must have the **TrustZone** option configured as **From Port**. All other APB ports on that AMIB must also be configured to be secure or non-secure.
- To configure an APB port as boot secure, all other APB ports in a group and the parent AMIB must be boot secure. The parent AMIB must also have the **TrustZone** option configured as **Boot Secure.**

2.2.3 Low-power interfaces, clock-gating

The AXI low-power interface, C channel, used in the hierarchical clock-gating feature contains the signals that Table 2-3 shows.

Table 2-3 AXI low-power interface

Signal	Direction	Source, destination	Description
CACTIVE	Output, input	Interconnect, controller	Interconnect active
CSYSREQ	Output, input	Controller, interconnect	System low-power request
CSYSACK	Output, input	Interconnect, controller	Low-power request acknowledgement

A low-power interface is present for each clock domain when hierarchical clock-gating is enabled. Hierarchical clock-gating is a global parameter in the NIC-400 configuration. Any slave interface that is configured as an AHB cannot support hierarchical clock gating completely because the protocol does not support it.

The AHB protocol expects a slave to take the address when issued. No mechanism exists for a slave to avoid this, so if the clock for an AHB interface was off, the address phase of the transfer is lost. Therefore, any AHB slave interface is required to be in its own unique clock domain.

To turn the AHB interface clock off, the system designer must ensure that no transactions are inhibited at this interface.

A **CACTIVE** output is provided to show the interface status.

The *AMBA AXI and ACE Protocol Specification* contains additional information on the function of these signals.

Note

You can treat the *AXI AMBA AXI and ACE Protocol Specification* description of the concept of **CACTIVE** being HIGH when **CSYSACK** falls as a denial of the request, with the clock continuing to run. It is not necessary to support this functionality when implementing clock-gating. When **CSYSACK** falls, it is always safe to gate the clock.

AMIBs with APB connections into another clock domain do not support hierarchical clock gating on that boundary. This is because of the large overhead of creating clock control circuitry for effectively one side of an APB asynchronous bridge. Therefore the system designer is responsible for ensuring that the clock is already enabled on this interface. The designer can achieve this in one of two ways:

- Firstly by ensuring by means external to the NIC that clock to these APB interfaces is enabled whenever it is accessed.
- Alternatively, all methods of accessing these APB interface(s) might pass through this clock domain used by these interface(s) at a point before the APB AMIB.

These possible actions enable the NIC to wakeup and maintain the clock until the transaction is complete.

2.3 Operation

This section describes how the CoreLink NIC-400 Network Interconnect operates and contains the following subsections:

- *AXI3 and AXI4 protocol conversion*
- *Hierarchical clock-gating* on page 2-16
- *Upsizing data width function* on page 2-16
- *Downsizing data width function* on page 2-18
- *FIFO and clocking function* on page 2-20
- *Arbitration* on page 2-22
- *Cyclic Dependency Avoidance Schemes (CDAS)* on page 2-22
- *SAS* on page 2-23
- *Lock support* on page 2-23
- *TrustZone technology and security* on page 2-24
- *Remap* on page 2-26
- *Global ID and ID reduction* on page 2-28.

2.3.1 AXI3 and AXI4 protocol conversion

This section describes:

- *AXI4 to AXI3*.
- *AXI3 to AXI4*.

AXI4 to AXI3

AXI4 long bursts are split into multiple AXI3 bursts, up to a length of 16 beats as required. You can determine the number of output transactions by the formula shown in Figure 2-1:

$$\frac{\text{sum of the number of incoming bytes of an input transaction}}{\text{number of bytes of an output beat}}$$

Figure 2-1 Determining the number of output transactions

When more than one transaction is output, then each one is 16 beats apart from the last transaction, except for the last, which might not be of the same length.

AXI3 to AXI4

The maximum possible AXI4 output burst length is 16 beats, unless the downsizing function is also configured.

If there is a requirement to prevent AXI3 transactions creating long bursts, then you can configure a burst limiter function. The burst limiter requires a programmer interface to enable selection of burst limiting or no burst limiting, see registers for the ASIB, IB and AMIB in Chapter 3 *Programmers Model*.

Note

AXI4 does not support `Lock`, therefore you must only use exclusive atomic accesses to access an AXI4 slave from an AXI3 master.

2.3.2 Hierarchical clock-gating

Hierarchical clock-gating is a feature that enables a system to transition to another power state. This can be a low-power state where in low activity scenarios, the power that the clock tree consumes can be saved. Hierarchical clock-gating enables an external clock controller to individually request clock domains in the interconnect to block new transactions from entering the interconnect when there are no outstanding transactions within the clock domain. The domain then acknowledges that this process is complete and the clock controller is able to remove the clock. Giving control over individual clock domains permits flexible system design and therefore flexible power state design.

The programmers model is distributed throughout the interconnect and therefore generally through multiple clock domains. See Chapter 3 *Programmers Model*. When the hierarchical clock-gating feature is enabled, and more than one clock domain contains the programmers view registers or access point, an additional clock domain is added to the interconnect. This clock domain distributes accesses to the programmers model between the user-specified clock domains. It automatically requests the clock for other domains and makes clock-gating transparent to the user when accessing the programmers model. This additional clock domain also has an AXI low-power interface that must be connected to a clock controller in the same way as the other interfaces. All communication between clock domains is carried out asynchronously so the clock frequency of this central ring can be set as the user requires, within the limits of bridge limitations stated in the documentation.

2.3.3 Upsizing data width function

The upsizer function can expand the data width by the following ratios:

- 1:2
- 1:4
- 1:8.

Upsizing only packs write data for write or read transactions that are cacheable. This section describes the packing rules for different burst types and acceptance capabilities, and the following definitions apply:

- an aligned input burst means that the address is aligned to the output data width word boundary, after the network aligns it to the size of the transfer
- an unaligned input burst means that the network does not align the address to the output data width word boundary, even after it aligns it to the size of the transfer
- if a transaction passes through, this means that the upsizer function does not change the input transaction size and type.

Note

- If the network splits input exclusive transactions into more than one output bus transaction, it removes the exclusive information from the multiple transactions it creates.
- If multiple responses from created transactions are combined into one response, then the order of priority is:
  - DECERR is the highest priority
  - SLVERR is the next highest priority
  - OKAY is the lowest priority.

In the examples in this section, the input data width is 64-bit, and the output data width is 128-bit, unless otherwise stated. This section describes:

- *INCR bursts* on page 2-17

- *Upsize*
- *WRAP bursts* on page 2-18
- *Fixed bursts* on page 2-18
- *Bypass merge* on page 2-18
- *Acceptance capability* on page 2-18.

INCR bursts

The network converts all input INCR bursts that complete within a single output data width into an INCR1 of the minimum SIZE possible, and it packs all INCR bursts into INCR bursts of the optimum size possible. Table 2-4 shows how the network converts INCR bursts when it upsizes them.

Table 2-4 Conversion of INCR bursts by the upsizer function

INCR burst type	Converted to
64-bit INCR1	Passes through unconverted
64-bit aligned INCR2	INCR1
8-bit aligned INCR8	INCR1, 64-bit
8-bit unaligned, byte address 1, 2, or 3, INCR5	INCR1, 128-bit
8-bit unaligned, byte address 4, 5, 6, or 7, INCR5	INCR2, 64-bit
64-bit unaligned INCR2	Passes through unconverted
64-bit aligned INCR4	INCR2
64-bit unaligned INCR4	Sparse INCR3

Note

Bursts are never merged.

Upsize

When upsizing, and protocol converting to or from AXI3 and AXI4, then the following maximum INCR burst lengths are shown in Table 2-5.

Table 2-5 Maximum INCR burst lengths

Ratio	AXI3 to AXI3	AXI3 to AXI4	AXI4 to AXI4	AXI4 to AXI3
1:2	Maximum Len16 output	Maximum Len16 output	Maximum Len256 output	Maximum Len16 output
1:4	Maximum Len16 output	Maximum Len16 output	Maximum Len256 output	Maximum Len16 output
1:8	Maximum Len16 output	Maximum Len16 output	Maximum Len256 output	Maximum Len16 output

WRAP bursts

All WRAP bursts are either passed through unconverted as WRAP bursts, or converted to one or two INCR bursts of the output bus. Table 2-6 shows how the network converts WRAP bursts when it upsizes them from 64-bit to 128-bit, that is, a ratio of 1:2.

Table 2-6 Conversion of WRAP bursts by the upsizer function

WRAP burst type	Converted to
128-bit aligned WRAP2	INCR1
128-bit aligned WRAP4	WRAP2
128-bit unaligned WRAP4	Depending on the address: • INCR2 + INCR1 • INCR1 + INCR2

Note

The network converts input WRAP bursts with a total payload that is less than the output data width to a single INCR.

Fixed bursts

All FIXED bursts pass through unconverted.

Bypass merge

You can configure the upsizer function to have a programmable bit named `bypass_merge`. If `bypass_merge` is asserted, the network does not alter any transactions that can pass through legally without alteration.

Acceptance capability

You can configure the upsizer to support 1-32 read transactions and 1-32 write transactions. The issuing capability is a maximum of twice the acceptance capability.

2.3.4 Downsizing data width function

The downsizer function reduces the data width by the following ratios:

- 2:1
- 4:1
- 8:1.

The downsizer does not merge data narrower than the destination bus if the transaction is marked as non-cacheable.

This section describes the following:

- *INCR bursts* on page 2-19
- *Downsize* on page 2-19
- *WRAP bursts* on page 2-19
- *FIXED bursts* on page 2-20
- *Bypass merge* on page 2-20
- *Acceptance capability* on page 2-20.

INCR bursts

The CoreLink NIC-400 Network Interconnect converts INCR bursts that fall within the maximum payload size of the output data bus to a single INCR. It converts INCR bursts that are greater than the maximum payload size of the output data bus to multiple INCR bursts. Table 2-7 shows how the network converts INCR bursts when it downsizes them.

Table 2-7 Conversion of INCR bursts by the downsizer function

INCR burst type	Converted to
Aligned INCR4	INCR8
Unaligned INCR4	INCR7[a]
Aligned INCR9	INCR16 + INCR2

a. This is only valid if the address is aligned to the destination width, and is not aligned to the source width. For example, if `0x4` is placed on a 64-32 bit downsizer, then `0x1` still requires an INCR8.

INCR bursts with a size that matches the output data width pass through unconverted.

The CoreLink NIC-400 Network Interconnect packs INCR bursts with a SIZE smaller than the output data width to match the output width whenever possible, using the upsizer transfer function. See *Upsizing data width function* on page 2-16.

Downsize

When downsizing, and protocol converting to or from AXI3 and AXI4, then the following maximum INCR burst lengths are shown in Table 2-5 on page 2-17.

Table 2-8 Maximum INCR burst lengths

Ratio	AXI3 to AXI3	AXI3 to AXI4	AXI4 to AXI4	AXI4 to AXI3
1:2	Maximum Len16 output	Maximum Len16 output	Maximum Len256 output[a]	Maximum Len16 output
1:4	Maximum Len16 output	Maximum Len16 output	Maximum Len256 output	Maximum Len16 output
1:8	Maximum Len16 output	Maximum Len16 output	Maximum Len256 output	Maximum Len16 output

a. In AXI3 to AXI4 downsizing, you can create long bursts from an AXI3 burst input. If a restriction to shorts bursts is required, you can control this by a programmable register. For more information, see the registers for the ASIB, IB and AMIB in Chapter 3 *Programmers Model*.

WRAP bursts

The CoreLink NIC-400 Network Interconnect always converts WRAP bursts to WRAP bursts of twice the length, up to the output data width maximum size of WRAP16, in which case, it treats the WRAP burst as two INCR bursts that can each map onto one or more INCR bursts.

Note

If a wrap transaction is aligned to the wrap boundary, it is converted into an INCR transaction.

FIXED bursts

The CoreLink NIC-400 Network Interconnect converts FIXED bursts to one or more INCR1 or INCRn bursts depending on the downsize ratio. Table 2-9 shows how the network converts FIXED bursts when it downsizes them.

Table 2-9 Conversion of FIXED bursts by the downsizer function

FIXED burst type	Converted to
FIXED1	INCR2
FIXED2	INCR2 + INCR2 + ...

The CoreLink NIC-400 Network Interconnect optimizes unaligned fixed bursts. If an unaligned input fixed burst maps onto a single output beat, then the output is a fixed burst of the optimal size.

Bypass merge

You can configure the downsizer function to have a programmable bit named `bypass_merge`. If `bypass_merge` is asserted, the network does not perform any packing of beats to match the optimum SIZE, up to the output data width SIZE.

An aligned input burst means that the address is aligned to the input data width word boundary after the network aligns it to the transfer size. An unaligned input burst means that the address is not aligned to the input data width word boundary, even after the network aligns it to the transfer size.

If a transaction passes through, this means that the downsizer function does not change the input transaction size and type.

Note

- If an exclusive transaction is split into multiple transactions at the output of the downsizer, the exclusive flag is removed and the master never receives an EXOKAY response. Response priority is the same as for the upsizer function. See *Upsizing data width function on page 2-16*.
- In the following example, the input data width is 128-bit and the output data width is 64-bit unless otherwise stated.

Acceptance capability

You can configure the acceptance capability to 1-32 read transactions and 1-32 write transactions. The maximum issuing capability is (ratio x acceptance capability +1).

2.3.5 FIFO and clocking function

If you configure the network as a clock frequency crossing bridge, then a FIFO function is also configured.

Note

You can configure the buffering for multiple outstanding transactions even if you are using a 1:1 clocking ratio.

You can instantiate a FIFO on any channel. You can configure the FIFO to implement both buffering and clock domain crossing functionality. You can define the FIFO to be:

- SYNC 1:1
- SYNC 1:n
- SYNC n:1
- ASYNC
- SYNC m:n

Note

You can dynamically change this through the GPV.

The network automatically determines that the width of the FIFO is the width of the widest payload, in or out of the block. You can configure the depth of the FIFO to be 2-32.

All clock boundary crossings are implemented using a FIFO structure with appropriate synchronization for the current mode of operation.

Changing the synchronization when you select programmable mode

You can change the boundary type by modifying the synchronization that is applied to the two pointers as they pass between domains. This ensures that the data in the FIFO is stable and safe to use.

To change the clocks, the synchronization must remain correct at all times. Table 2-10 shows the actions you must take to convert from one mode to another.

Table 2-10 How to change modes

Original mode	Required mode	Action
ASYNC	Any other mode	Change the clocks then change the register.
Any mode	ASYNC	Change the register then change ASYNC. BRESP from the GPV implies that the update is complete.
SYNC m:n	SYNC 1:1	Change the clocks, then change the register.
SYNC 1:1	SYNC m:n	Change the register, then change the clocks.

Note

For some changes, it is necessary to use a different setting, that is, you can only change safely from SYNC 1:n to SYNC m:1 by first programming the register to SYNC m:n, before the clock update.

Data release mechanism

When you configure a write data FIFO of at least 4, you can also set an additional write tidemark function, named `wr_tidemark`. This is a tidemark level that stalls the release of the transaction until:

- The network receives the WLAST beat.
- The write FIFO becomes full.
- The number of occupied slots in the write data FIFO exceeds the write tidemark. See Chapter 3 *Programmers Model*.

2.3.6 Arbitration

You can program the arbitration algorithm for all arbitration nodes within the infrastructure.

At the entry point to the infrastructure, all transactions are allocated a local QoS that you can configure to be:

- static
- programmable
- received from the attached master, for AXI only.

The arbitration of the transaction throughout the infrastructure uses this QoS. See Chapter 3 *Programmers Model*.

At any arbitration node, a fixed priority exists for transactions with a different QoS. The highest value has the highest priority. If there are coincident transactions at an arbitration node with the same QoS that require arbitration, then the Network uses a *Least Recently Used* (LRU) algorithm.

Note

Alternatively, you can source the allocated QoS from the QoS-400 add-on.

2.3.7 Cyclic Dependency Avoidance Schemes (CDAS)

Because the AXI protocol permits re-ordering of transactions, it might be necessary for the CoreLink NIC-400 Network Interconnect to enforce rules to prevent deadlock when routing multiple transactions concurrently to multiple slaves from a point of ingress to the interconnect, that is, at a slave interface.

Each ASIB can have a different CDAS configured. The same CDAS scheme is configured for both read and write transactions, but they operate independently.

This section describes:

- *Single slave*
- *Single slave per ID*.

Single slave

This ensures that at a ASIB:

- all outstanding read transactions are to a single end destination
- all outstanding write transactions are to a single end destination.

If the slave interface receives a transaction to a different destination than to the current destination for that transaction type, the network stalls the transactions until all the outstanding transactions of that type are completed.

Single slave per ID

This ensures that at a ASIB:

- all outstanding read transactions with the same ID go the same destination
- all outstanding write transactions with the same ID go the same destination.

When the ASIB receives a transaction:

- if it has an ID that does not match any outstanding transactions, it passes the CDAS

- if it has an ID that matches the ID of an outstanding transaction, and the destinations also match, it passes the CDAS
- if it has an ID that matches the ID of an outstanding transaction, and the destinations do not match, it fails the CDAS check and is stalled.

A stalled transaction remains stalled until one of the rules passes.

The AMBA Designer tool automatically detects when this is required. See *Additional reading* on page vii.

Depending on the configured topology and ASIB CDAS scheme, there is still a possibility for a cyclic dependency deadlock because of the AW and W channel ordering rules. This is detected by AMBA Designer and indicated by a loop error.

You can resolve this by either changing:

- the configuration topology
- all but one of the ASIB CDAS schemes that feed into the loop to a single slave
- a single switch slave interface on the loop to a *Single Active Slave* (SAS).

2.3.8 SAS

SAS enforces the rule that at a divergent slave interface of a switch, that an AW address beat is stalled provided that there are any outstanding write data beats to a different master interface of the switch.

2.3.9 Lock support

You set support for locked transaction for AXI3 masters and slaves at configuration time. The AXI4 protocol does not support locked transactions. The CoreLink NIC-400 Network Interconnect infrastructure is configured for lock support into all switch master interfaces that are required to provide lock support for all the relevant masters and slaves in the system. See *Lock transactions* on page 2-7 for information on the scope of lock support for AHB-Lite slave interfaces.

At a switch master interface with lock support, logic exists that:

- Stalls a locked transaction after it is arbitrated.

  Note

  If a co-incident transaction exists on the other address channel, it is not stalled unless it is a lock transaction.

- Stalls the other address channel.
- Permits all the outstanding transactions to complete.
- Enables the locking transactions source read and write channels when there are no outstanding transactions.
- Enables all sources for arbitration, and normal operation continues when an unlocking transaction completes.
- When the network receives a locking transaction, if there is a co-incident lock transaction on the other address channel, then the read always takes priority, and the write address transaction is stalled.

Note

The NIC supports lock functionality for 32-bit data beat accesses. You can lock beats of other sizes, but if they are upsized or downsized, it is possible that leading write data are output from the sizing function for the unlocking transaction before all the locked transactions have completed.

2.3.10 TrustZone technology and security

This section applies if you are building a system based on the secure and non-secure capabilities that TrustZone technology provides. If the system does not require security using TrustZone technology, configure all master interfaces to be non-secure.

This section contains the following subsections:

- *TrustZone scope*
- *Slave interface security* on page 2-25
- *Internal programmers view* on page 2-25
- *Security checking for master interfaces* on page 2-25.

TrustZone scope

The security checks that TrustZone technology implements cover the scope of a configured network.

Note

TrustZone is a brand name that represents aspects of implementing ARM Security Extensions.

For example, security checks that are not within the scope of the network are:

Physical attack

Physical attack on the device.

Non-TrustZone-aware masters being made secure

A master might require access to the *Global Programmers View* (GPV) and in this case, you can tie the security transaction indicator bits 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.

System implementation information

If you do not consider all the masters that have access to the GPV, this can produce security vulnerabilities. For example:

- If a non-secure state master can set QoS requirements effecting its non-secure transactions, then that non-secure state master can use this capability, in conjunction with traffic analysis, to determine the QoS and priority settings of a secure master. This can be a threat in particular implementations.
- A TrustZone-aware slave requires you to set the connecting network as non-secure so that the network does not filter the secure traffic and leaves the slave to determine the correct response. Consider the master that can make this non-secure configuration against and the master, or masters, that can program the TrustZone-aware slave.

Topology issues

It might be possible to suffer timing attacks because of the topology configuration you chose. For example, if two cascaded switches exist with a shared AXI link between them, then continuous non-secure accesses to a non-secure slave might block secure transactions to a different secure slave.

Resets It might be possible to carry out a secure attack by resetting only parts of a data path, whether it be a data path section in an individual clock domain within a network, or within a master or slave.

Hierarchical clock-gating

It might be possible to carry out a denial of service attack by gating clock domains. Only masters in the secure domain must access the clock controller.

Slave interface security

At configuration time, each slave interface, whether it belongs to the AXI or AHB protocol, has the following options for setting the security assignment of all its transactions:

- input from the external master, for AXI masters only
- tied-off to always issue transactions as secure
- tied-off to always issue transactions as non-secure.

Internal programmers view

The programmers view is always secure access only. Any non-secure transaction intended to access a register, input to a configuration, returns a DECERR, and no register access is provided.

Note

If you configure a dedicated configuration port to gain access to the GPV, then you must connect it to a secure master, or have a security check that is external to the CoreLink NIC-400 Network Interconnect.

Security checking for master interfaces

You can configure each master interface to be:

Always secure

The master rejects non-secure transactions.

Always non-secure

The master accepts both secure and non-secure transactions.

Boot secure You can use software to configure whether it permits secure and non-secure transactions to access components attached to this master using the Always secure and Always non-secure options above.

Note

- If you change the security of a master interface, the change does not occur simultaneously for all the masters in the system because of the distributed nature of the GPV.
- Outstanding transactions, or active lock sequences, underway within the network at the time of the security update use the old security settings for their security check.

For an APB master interface, where multiple slaves exist on a single interface, each APB slave has its own security check.

If an incoming transaction is non-secure, either because the slave interface is configured to be non-secure, or the input security bit is set be non-secure, then if that transaction is intended for a master interface that is currently secure, then that transaction is returned with a DECERR, and the transaction is not transferred to the slave.

All accesses must be secure to gain access to any programmers model register. Any non-secure accesses to the programmers model receive a DECERR response. See Chapter 3 *Programmers Model*.

Security registers are not updated if a pending transaction exists, or if a current ongoing lock sequence exists.

2.3.11 Remap

Registers in the programmers model control the remap functionality. See Table 3-4 on page 3-9 in Chapter 3 *Programmers Model* for more information.

You can define a number of remap states using eight bits of the remap register, and a bit in the remap register controls each remap state.

Note

You can use each remap state to control the address decoding for one or more slave interfaces. If a slave interface is affected by two remap states that are both asserted, the remap state with the lowest remap bit number takes precedence.

You can configure each slave interface independently so that a remap state can perform different functions for different masters.

A remap state can:

- alias a memory region into two different address ranges
- move an address region
- remove an address region.

Because of the nature of the distributed register sub-system, the masters receive the updated remap bit states in sequence, and not simultaneously.

A slave interface does not update to the latest remap bit setting until:

- the address completion handshake accepts any transaction that is pending
- any current lock sequence completes.

Note

The BRESP from a GPV after a remap update guarantees that the next transaction issued to each slave interface, or the first one after the completion of a locked sequence, uses the updated value.

Figure 2-2 on page 2-27 to Figure 2-6 on page 2-28 show examples of how different remap states interact with each other. Consider a configuration that uses three remap bits. Figure 2-2 on page 2-27 shows the memory map when remap is set to 000, representing no remap.

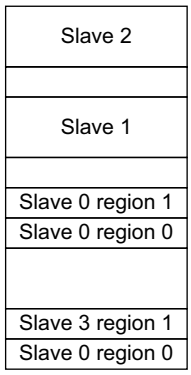

Figure 2-2 No remap, remap set to 000

This has a default memory map that divides slave 0 and slave 3 into two separate regions. At power-up, slave 0 region 0 is aliased over slave 3 region 0. After power-up, the slave 0 region 0 alias can be removed as Figure 2-3 shows.

Figure 2-3 Remap set to 001

Alternatively, you can move slave 1 to the bottom of the address range by setting remap to 010 as Figure 2-4 shows.

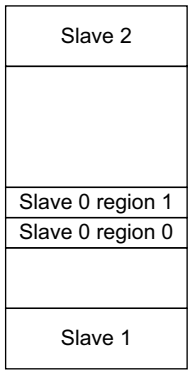

Figure 2-4 Remap set to 010

Note

Remap bit 0 still takes precedence if you set it as Figure 2-5 on page 2-28 shows.

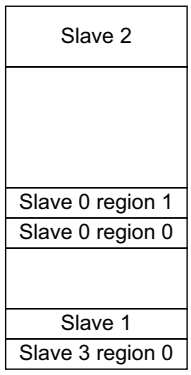

Figure 2-5 Remap set to 011

In addition, you can remove memory regions entirely. Figure 2-6 shows that if you set remap to 101, Slave 1 is removed.

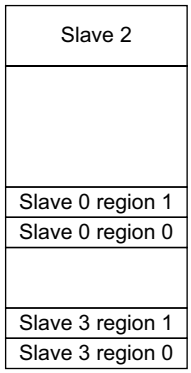

Figure 2-6 Remap set to 101

2.3.12 Global ID and ID reduction

The CoreLink Network Interconnect uses a fixed width ID for all transactions passing through the network. This enables you to construct complex topologies.

The global ID contains the following sections:

***Interconnect ID* (IID)**

IID is the number that identifies the interconnect connected to the ASIB.

***Virtual ID* (VID)**

A master that is connected to an ASIB supplies the VID bits.

***Slave Interface ID* (SIID)**

The ASIB uses the SIID to identify the slave interface in the switch that was the source of the transaction. AMBA Designer assigns the SIID.

In many cases the ID width of a master interface can be reduced to a minimum value through optimization. You can achieve this through a selectable GUI feature known as ID reduction, that automatically performs the procedure.

2.4 Optional features

If you have licensed the appropriate product you can include the following additional services that are extensions to the CoreLink NIC-400 Network Interconnect:

- QoS
- QVN
- TLX.

2.4.1 QoS

Using this service provides:

- programmable QoS facilities for attached AMBA masters
- regulation of read and write requests
- configurable QoS options for ASIB and IB
- low gate count
- low power consumption
- no cycles of latency added to requests when inactive.

For more information, see *CoreLink QoS-400 Network Interconnect Advanced Quality of Service, Supplement to CoreLink NIC-400 Network Interconnect Technical Reference Manual*.

2.4.2 QVN

Using this service:

- Prevents congestion between traffic flows in the system, by enabling the system designer to separate traffic flows with conflicting requirements on to different virtual networks. For example high bandwidth bus traffic sources can be prevented from blocking the flow of latency critical bus traffic.
- Enables you to configure up to eight virtual networks. However, you can only have a maximum of four virtual networks on any single master or slave interface.
- Enables you to configure virtual network assignment by an addressable path, from masters to slaves.
- Enables you to configure the slave interfaces to mask the resource allocation latency of virtual networks.

For more information, see *CoreLink QVN-400 Network Interconnect Advanced Quality of Service for Virtual Networks, Supplement to CoreLink NIC-400 Network Interconnect Technical Reference Manual.*

2.4.3 TLX

Using this service:

- Provides a mechanism to reduce the number of signals in an AXI point-to-point connection and enables it to be routed over a longer distance.
- You can configure the TLX as an:
  - AHB to AXI3 bridge
  - AHB to AXI4 bridge
  - AXI3 to AHB bridge
  - AXI4 to AHB bridge

  - AXI3 to AX13 bridge
  - AXI4 to AXI4 bridge
  - AXI4 to AXI3 bridge
  - AXI3 to AXI4 bridge.
- Enables you to work in conjunction with QVN.
- A Thin Link implementation is partitioned into two layers:
  - *Data Link Layer* (DLL)
  - *Physical Layer* (PL).
- Enables you to use all existing options for an ASIB, AMIB, and IB when the thin link option is selected.

For more information, see *CoreLink TLX-400 Network Interconnect Thin Links, Supplement to CoreLink NIC-400 Network Interconnect Technical Reference Manual*.

Chapter 3
Programmers Model

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

- *About the programmers model* on page 3-2
- *Configuration programmers model* on page 3-3.

3.1 About the programmers model

This chapter describes the architecture of the CoreLink NIC-400 Network Interconnect AMBA infrastructure component. It describes the programmers interface and system characteristics.

3.2 Configuration programmers model

The CoreLink NIC-400 Network Interconnect contains configuration registers, partitioned into a number of individual 4KB blocks that you can program using the *Global Programmers View* (GPV). The base address of each GPV region is set at configuration time in CoreLink ADR-400 AMBA Designer.

You can configure any slave interface to have access to all of the registers in the programmers view.

The following restrictions apply to accessing the GPV. The GPV:

- Only supports AXI transactions of **AxSIZE** equal to 32.
- Only supports secure transactions.

  Note

  AxPROT[1] must always be 1'b0 to denote a secure transfer.

- Does not support interleaved **WDATA**.
- Only supports aligned transactions. It does not support unaligned accesses.
- Does not support sparsely strobed write data beats. The product only guarantees to support fully strobed or strobeless 32-bit write data beats.

Note

Any registers that a switch requires are implemented within the register block of the associated *Interface Block* (IB). If no IB is attached, then you can configure an IB to specifically provide programmable registers.

Ensure that you access the GPV using non-cacheable transactions.

Note

Before you access the GPV, you must ensure that all clocks are running.

3.2.1 Register block types

The following types of register block exist:

- one register block for each CoreLink NIC-400 Network Interconnect configuration
- one register block for each IB, where the IB can be:
  - *AXI Slave Interface Block* (ASIB), see Table 3-1 on page 3-4
  - *AXI Master Interface Block* (AMIB), see Table 3-2 on page 3-6
  - AXI internal network *Interface Block* (IB), see Table 3-3 on page 3-7.

Figure 3-1 on page 3-4 shows the address map of the programmers model. It contains one fixed base address, and all the other programmers model 4KB blocks are stacked.

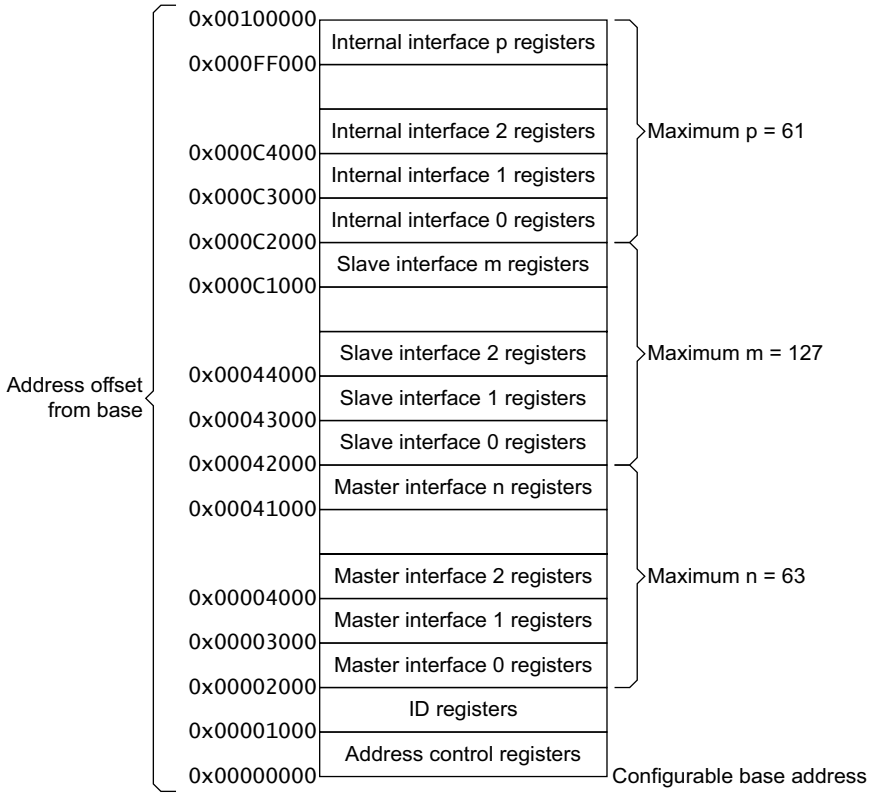

Figure 3-1 Address map of the programmers model

The base address of a register block is determined by the node number assigned to it. There is no requirement for register blocks to be contiguous.

The type defines the number of register blocks in a single CoreLink NIC-400 Network Interconnect configuration. Table 3-1, Table 3-2 on page 3-6, and Table 3-3 on page 3-7 show the register block sub-types for each of the main types.

Table 3-4 on page 3-9 shows the address region control registers and Table 3-5 on page 3-10 shows the peripheral ID registers.

Note

In Table 3-1 to Table 3-5 on page 3-10, reserved means:

- read as zeros
- writes are ignored.

AHB only means that this register is interpreted as reserved if the interface is not AHB.

Table 3-1 shows the registers that exist for each ASIB.

Table 3-1 Registers for each ASIB

Address offset	Type	Width	Reset value	Name	Description
0x000	-	-	-	-	Reserved.
0x004	-	-	-	-	Reserved.
0x008	-	-	-	-	Reserved.

Table 3-1 Registers for each ASIB (continued)

Address offset	Type	Width	Reset value	Name	Description
0x00C	-	-	-	-	Reserved.
0x020	RW	3	4	sync_mode	This register is only present if the clock relationship across the ASIB is defined as programmable before RTL generation. You can configure the register bits as follows: **0** sync 1:1. **1** sync m:1. **2** sync 1:n. **3** sync m:n. **4** async. **5** reserved. **6** reserved. **7** reserved.
0x024	RW	1	0	fn_mod2	Bypass merge. This register is only present if upsizing or downsizing, see *Upsizing data width function* on page 2-16, *Downsizing data width function* on page 2-18, and *Bypass merge* on page 2-18.
0x028	RW	3	0	fn_mod_ahb	This register is valid for AHB interfaces only. You can configure the register bits as follows: **0** rd_incr_override. **1** wr_incr_override. **2** lock_override. See *Lock transactions* on page 2-7 for information on overriding locks. See *Combination 4* on page 2-6 for information on wr_incr_override and rd_incr_override.
0x02C - 0x03C	-	-	-	-	Reserved.
0x040	RW	4	[a]	wr_tidemark	Valid only with a FIFO for the WFIFO channel, where the tidemark value is configured to a non-zero value before RTL generation. See *FIFO and clocking function* on page 2-20 for information on wr_tidemark.
0x044 - 0x0FC	-	-	-	-	Reserved.
0x100	RW	4	0[b]	read_qos	Read channel QoS value. **Note** This register is only present when the QoS settings for an ASIB (master) have been set to programmable.
0x104	RW	4	0[b]	write_qos	Write channel QoS value. **Note** This register is only present when the QoS settings for an ASIB (master) have been set to programmable.
0x108	RW	2	0	fn_mod	Issuing functionality modification register. This register sets the block issuing capability to one outstanding transaction. You can configure the register bits as follows: **0** Read issuing, read_iss_override. **1** Write issuing, write_iss_override.

Table 3-1 Registers for each ASIB (continued)

Address offset	Type	Width	Reset value	Name	Description
0x10C	-	-	-	-	Reserved.
0x110	R/W	1	0	fn_mod_bb	This register is only present when downsizing to AXI4 is required. Long burst functionality modification register. This controls burst breaking of long bursts as follows: **0** Long bursts cannot be generated at the output of the ASIB. **1** Long bursts can be generated at the output of the ASIB. **Note** If the programmers view is not turned on then the default value is used, and long bursts are not generated.
0x114 - 0xFFC	-	-	-	-	Reserved.

a. The reset value is initialized to the tidemark value that you set in the configuration GUI in CoreLink ADR-400 AMBA Designer.

b. If you set the QoS Type parameter to Programmable, you can set the reset value of this register yourself.

Table 3-2 shows the registers that exist for each IB.

Table 3-2 Registers for each IB

Address offset	Type	Width	Reset value	Name	Description
0x000	-	-	-	-	Reserved.
0x004	-	-	-	-	Reserved.
0x008	RW	2	0	fn_mod_bm_iss	Bus matrix issuing functionality modification register. This register is only present if the block is connected directly to a switch. This register sets the issuing capability of the preceding switch arbitration scheme to 1. You can configure the register bits as follows: **0** Read issuing, read_iss_override. **1** Write issuing, write_iss_override.
0x00C	-	-	-	-	Reserved.
0x020	RW	3	4	sync_mode	This register is only present if the clock relationship across the IB is defined as programmable before RTL generation. You can configure the register bits for the following clock domain boundaries: **0** sync 1:1. **1** sync m:1. **2** sync 1:n. **3** sync m:n. **4** async. **5** reserved. **6** reserved. **7** reserved.
0x024	RW	1	0	fn_mod2	Bypass merge. This register is only present if upsizing or downsizing. See *Upsizing data width function* on page 2-16, *Downsizing data width function* on page 2-18, and *Bypass merge* on page 2-18.

Table 3-2 Registers for each IB (continued)

Address offset	Type	Width	Reset value	Name	Description
0x028	-	-	-	-	Reserved.
0x02C	R/W	1	0	fn_mod_bb	This register is only present when downsizing to AXI4 is required. Long burst functionality modification register. This controls burst breaking of long bursts as follows: **0** Long bursts cannot be generated at the output of the IB. **1** Long bursts can be generated at the output of the IB. **Note** If the programmers view is not turned on then the default value is used, and long bursts are not generated.
0x030 - 0x03C	-	-	-	-	Reserved
0x040	RW	4	a	wr_tidemark	Valid only with a FIFO for the WFIFO channel, where the tidemark value has been configured to a non-zero value prior to RTL generation.
0x044	-	-	-	-	Reserved.
0x100	-	-	-	-	Reserved.
0x104	-	-	-	-	Reserved.
0x108	RW	2	0	fn_mod	Issuing functionality modification register. Issuing override, sets block issuing capability to one transaction and you can configure the bits as follows: **0** Read issuing, read_iss_override. **1** Write issuing, write_iss_override.
0x10C	-	-	-	-	Reserved.
0x114 - 0xFFC	-	-	-	-	Reserved.

a. The reset value is initialized to the tidemark value that you set in the configuration GUI in CoreLink ADR-400 AMBA Designer.

Table 3-3 shows the registers that exist for each AMIB.

Table 3-3 Registers for each AMIB

Address offset	Type	Width	Reset value	Name	Description
0x000	-	-	-	-	Reserved.
0x004	-	-	-	-	Reserved.
0x008	RW	2	0	fn_mod_bm_iss	Bus matrix issuing functionality modification register. This register is only present if the block is connected directly to a switch. This register sets the issuing capability of the preceding switch arbitration scheme to 1. You can configure the register bits as follows: **0** Read issuing, read_iss_override. **1** Write issuing, write_iss_override.
0x00C	-	-	-	-	Reserved.

Table 3-3 Registers for each AMIB (continued)

Address offset	Type	Width	Reset value	Name	Description
0x020	RW	3	4	sync_mode	This register is only present if the clock relationship across the AMIB is defined as being programmable prior to RTL generation. You can configure the register bits to create different clock domain boundaries as follows: **0** sync 1:1. **1** sync m:1. **2** sync 1:n. **3** sync m:n. **4** async. **5** reserved. **6** reserved. **7** reserved.
0x024	RW	1	0	fn_mod2	Bypass merge. This register is only present if upsizing or downsizing. See *Upsizing data width function* on page 2-16 and *Downsizing data width function* on page 2-18.
0x028	-	-	-	-	Reserved.
0x02C	R/W	1	0	fn_mod_bb	This register is only present when downsizing to AXI4 is required. Long burst functionality modification register. This controls burst breaking of long bursts as follows: **0** Long bursts cannot be generated at the output of the AMIB. **1** Long bursts can be generated at the output of the AMIB. **Note** If the programmers view is not turned on then the default value is used, and long bursts are not generated.
0x030 - 0x03C	-	-	-	-	Reserved.
0x040	RW	4	a	wr_tidemark	Valid only with a FIFO for the WFIFO channel, where the tidemark value has been configured to a non-zero value prior to RTL generation.
0x044	RW	2	0	ahb_cntl	This register is available for AHB only. You can configure the register bits as follows: **0** decerr_en. **1** force_incr. See *AHB master interfaces* on page 2-10.
0x100 - 0x104	-	-	-	-	Reserved.
0x108	RW	2	0	fn_mod	Issuing functionality modification register. This register is only available if you are upsizing or downsizing, or you have a FIFO for any of the channels. This register sets the block issuing capability to be forced to one transaction. You can configure the register bits as follows: **0** Read issuing, read_iss_override. **1** Write issuing, write_iss_override.
0x10C - 0xFFC	-	-	-	-	Reserved.

a. The reset value is initialized to the tidemark value that you set in the configuration GUI in CoreLink ADR-400 AMBA Designer.

3.2.2 Register blocks

This section contains the following subsections:

- *Address region control*
- *Peripheral ID registers* on page 3-10.

Address region control

Table 3-4 shows the address region control registers.

Table 3-4 Address region control registers

Address offset	Type	Width	Reset value	Name	Description
0x0	WO	8	0x00	remap	Remap register. Up to eight global remap states are available.
0x4	WO	-	-	-	Reserved.
0x08	WO	1 - 16	0x0	security0	Slave 0 security setting. This consists of one bit for non-virtual slaves, and up to 16 bits for virtual or APB master interfaces, and you can configure the register bits as follows: **0** Secure. **1** Non-secure. **Note** • For virtual or APB master interfaces with 16 security setting bits, each bit position maps onto the region number. For example, the security1[5] bit is the security setting for the address region for master interface node number 2, region 5. • You can identify the correct master interface node number through using AMBA Designer during configuration.

Table 3-4 Address region control registers (continued)

Address offset	Type	Width	Reset value	Name	Description
0x0C	WO	1 - 16	0x0	security1	Slave 1 security setting. This consists of one bit for non-virtual slaves, and up to 16 bits for virtual or APB master interfaces, and you can configure the register bits as follows: **0** Secure. **1** Non-secure. **Note** • For virtual or APB master interfaces with 16 security setting bits, each bit position maps onto the region number. For example, the security1[5] bit is the security setting for the address region for master interface node number 31, region 5. • You can identify the correct master interface node number through using AMBA Designer during configuration.
0x10 - 0x104	WO	1 - 16	0x0	security<*n*>	Slave n security setting. It contains one bit for non-virtual slaves, and up to 16 bits for APB master interfaces and you can configure the register bits as follows: **0** Secure. **1** Non-secure. **Note** • For virtual or APB master interfaces with 16 security setting bits, each bit position maps onto the region number. For example, the security1[5] bit is the security setting for the address region for master interface node number <node> + 2, region 5. • You can identify the correct master interface node number through using AMBA Designer during configuration.
0x110 - 0xFFF	RO	-	-	-	Reserved.

A configuration can contain a maximum of 64 security registers, that is, $1 < n < 64$. Therefore, if the configuration contains 64 master interfaces, then register security 63 is 0x104. These registers are write-only because they are global accesses on the GPV.

Peripheral ID registers

If you configure any registers in the programmers view, peripheral ID registers are always created. This provides a low gate count option for identification. Table 3-5 shows the peripheral ID registers.

Table 3-5 Peripheral ID registers

Address offset	Type	Width	Reset value	Name	Description
0x0 - 0xFCC	RO	-	-	-	Reserved
0xFD0	RO	8	0x04	Peripheral ID4	4KB count, JEP106 continuation code
0xFD4	RO	8	0x00	Peripheral ID5	Reserved
0xFD8	RO	8	0x00	Peripheral ID6	Reserved

Table 3-5 Peripheral ID registers (continued)

Address offset	Type	Width	Reset value	Name	Description
0xFDC	RO	8	0x00	Peripheral ID7	Reserved
0xFE0	RO	8	0x00	Peripheral ID0	Part Number [7:0]
0xFE4	RO	8	0xB4	Peripheral ID1	JEP106[3:0], part number [11:8]
0xFE8	RO	8	0x0B	Peripheral ID2	Revision, JEP106 code flag, JEP106[6:4]
0xFEC	RO	8	0x00	Peripheral ID3	You can set this using the AMBA Designer *Graphical User Interface* (GUI)
0xFF0	RO	8	0x0D	Component ID0	Preamble
0xFF4	RO	8	0xF0	Component ID1	Generic IP component class, preamble
0xFF8	RO	8	0x05	Component ID2	Preamble
0xFFC	RO	8	0xB1	Component ID3	Preamble

Appendix A
Signal Descriptions

This appendix describes the signal conventions used by NIC-400. It contains the following sections:

- *Global signals* on page A-2
- *Signal direction* on page A-3
- *AXI3 and AXI4 signals* on page A-4
- *APB signals* on page A-9
- *AHB-Lite signals* on page A-10
- *QVN signals* on page A-14.

A.1 Global signals

Table A-1 shows the global NIC-400 signals. These signals are used by all the protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-1 Global signals

NIC-400 signal adaptation[a]	Source	Description
<Domain name>clk	Clock source	Global clock signal
<Domain name>resetn	Reset source	Global reset signal
<Domain name>clken	Enable source	Global enable signal
<Domain name>clk_r	Clock source	GPV clock signal
<Domain name>reset_r	Reset source	GPV reset signal

a. <Domain name> is a name that you create when using AMBA Designer.

A.1.1 Low-power interface signals

Table A-2 shows the signals of the optional low-power interface. These signals are used by all protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-2 Read data channel signals

NIC-400 signal adaptation[a]	Source	Description
csyreq_cd_<Domain name>	Clock controller	System exit low-power state request
cysack_cd_<Domain name>	Peripheral device	Exit low-power state acknowledgement
cactive_cd_<Domain name>	Peripheral device	Clock active

a. <Domain name> is a name that you create when using AMBA Designer.

A.2 Signal direction

The signals described in the remaining sections can be implemented as inputs or outputs. Figure A-1 provides an example of the **awid_<port_name>** signal, and the **awid_<port_name>_s** or **awid_<port_ name>_m** signal described in Table A-3 on page A-4. The term **<port_name>** is a component part of the signal that is added by the system.

In a full NIC-400 system, the signal naming convention takes the name of the signal and the component part only, whilst if there is only a bridge connected to a single component, then the **s** and **m** extensions are added to the signal.

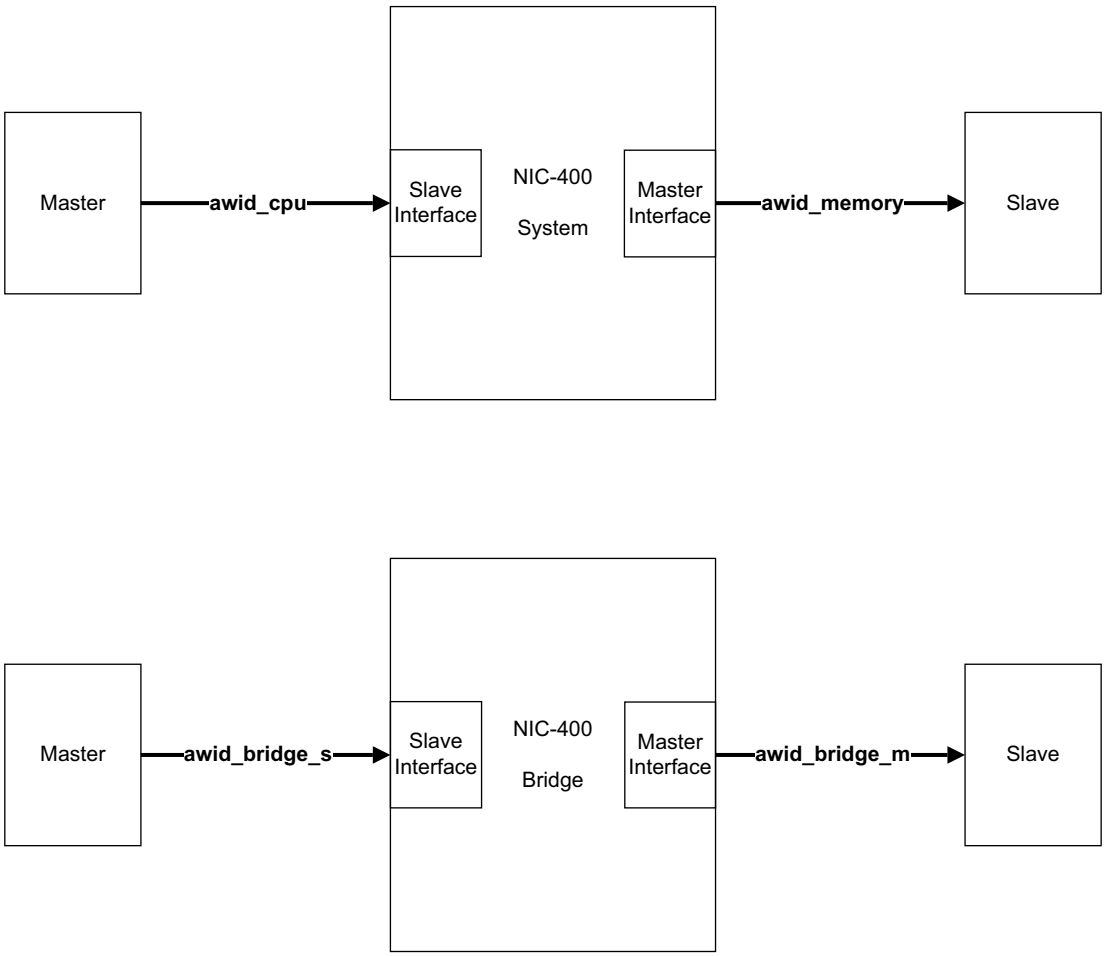

Figure A-1 Signal direction

For this reason the signal tables mostly reflect:

- Signal name
- Source
- Description.

A.3 AXI3 and AXI4 signals

The following signals are described:

- *Write address channel signals*
- *Write data channel signals* on page A-5
- *Write response channel signals* on page A-6
- *Read address channel signals* on page A-7
- *Read data channel signals* on page A-8.

A.3.1 Write address channel signals

Table A-3 shows the AXI write address channel signals. Unless the description indicates otherwise, these signals are used by AXI3 and AXI4 protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-3 Write address channel signals

AXI signal NIC-400 adaptation[a]	Source	Description
awid_x[b][n:0]	Master	Write address ID. Where *n* is a variable: • For an ASIB**, ID width** {0-16} where 0 indicates that no value has been selected. • For an AMIB, either: — **Global ID width** {1-24} or — if **ID reduction** has been selected, then the values can be viewed from the AMIBs ID reduction report from the GUI.
awaddr_x[b][n:0]	Master	Write address: Where: *n* is equal to address width -1. Address width is in the range of 32-64 bits.
awlen_x[b][3:0]	Master	Burst length for AXI3.
awlen_x[b][7:0]	Master	Burst length for AXI4.
awsize_x[b][2:0]	Master	Burst size.
awburst_x[b][1:0]	Master	Burst type.
awlock_x[b][1:0]	Master	Lock type for AXI3.
awlock_x[b]	Master	Lock type for AXI4.
awcache_x[b][3:0]	Master	Memory type.
awprot_x[b][2:0]	Master	Protection type.
awqos_x[b][3:0]	Master	Quality of Service, QoS. Only when enabled by the GUI.
awregion_x[b][3:0]	Master	Region identifier. Only when enabled by the GUI.

Table A-3 Write address channel signals (continued)

AXI signal NIC-400 adaptation[a]	Source	Description
awuser_x[b][n:0]	Master	User definable signal. Where: *n* is equal to user defined width -1. A user defined width is in the range of 0-256 bits.
awvalid_x[b]	Master	Write address valid.
awready_x[b]	Slave	Write address ready.

a. You can select uppercase or lowercase signal names from the GUI.

b. Where **x** is:

<port_name>_s for a bridge slave interface

<port_name>_m for a bridge master interface

<port_name> for a system slave interface or master interface.

A.3.2 Write data channel signals

Table A-4 shows the AXI write data channel signals. Unless the description indicates otherwise, these signals are used by AXI3 and AXI4 protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-4 Write data channel signals

AXI signal NIC-400 adaptation[a]	Source	Description
wid_x[b][n:0]	Master	Write address ID. Where *n* is a variable: • For an ASIB**, ID width** {0-16} where 0 indicates that no value has been selected. • For an AMIB, either: — **Global ID width** {1-24} or — if **ID reduction** has been selected, then the values can be viewed from the AMIBs ID reduction report from the GUI. **Note** This signal is not present in AXI4.
wdata_x[b][n:0]	Master	Write data. Where: *n* is equal to data width - 1. Data widths can be 32, 64, 128 or 256 bits.
wstrb_x[b][n:0]	Master	Write strobes When HIGH, specify the byte lanes of the data bus that contain valid information. There is one write strobe for each eight bits of the write data bus, therefore **wstrb[n]** corresponds to **wdata[(8n)+7: (8n)]**.
wlast_x[b]	Master	Write last.

Table A-4 Write data channel signals (continued)

AXI signal NIC-400 adaptation[a]	Source	Description
wuser_x[b]**[n:0]**	Master	User defined signal. Where: *n* is equal to user defined width -1. A user defined width is in the range of 0-256 bits. If 0 is selected then the signal is not used.
wvalid_x[b]	Master	Write valid.
wready_x[b]	Slave	Write ready.

a. You can select uppercase or lowercase signal names from the GUI.

b. Where **x** is:

<port_name>_s for a bridge slave interface

<port_name>_m for a bridge master interface

<port_name> for a system slave interface or master interface.

A.3.3 Write response channel signals

Table A-5 shows the AXI write response channel signals. Unless the description indicates otherwise, these signals are used by AXI3 and AXI4 protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-5 Write response channel signals

AXI signal NIC-400 adaptation[a]	Source	Description
bid_x[b]**[n:0]**	Slave	Response ID tag. Where *n* is a variable: • For an ASIB**, ID width** {0-16} where 0 indicates that no value has been selected. • For an AMIB, either: — **Global ID width** {1-24} or — if **ID reduction** has been selected, then the values can be viewed from the AMIBs ID reduction report from the GUI.
bresp_x[b]**[1:0]**	Slave	Write response.
buser_x[b]**[n:0]**	Slave	User defined signal. Where: *n* is equal to user defined width -1. A user defined width is in the range of 0-256 bits. If 0 is selected then the signal is not used.
bvalid_x[b]	Slave	Write response valid.
bready_x[b]	Master	Write response ready.

a. Where:

You can select uppercase or lowercase signal names from the GUI.

b. Where **x** is:

<port_name>_s for a bridge slave interface

<port_name>_m for a bridge master interface

<port_name> for a system slave interface or master interface.

A.3.4 Read address channel signals

Table A-6 shows the AXI read address channel signals. Unless the description indicates otherwise, these signals are used by AXI3 and AXI4 protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-6 Write address channel signals

AXI signal NIC-400 adaptation[a]	Source	Description
arid_x[b][n:0]	Master	Read address ID. Where *n* is a variable. • For an ASIB**, ID width** {0-16} where 0 indicates that no value has been selected. • For an AMIB, either: — **Global ID width** {1-24} or — if **ID reduction** has been selected, then the values can be viewed from the AMIBs ID reduction report from the GUI.
araddr_x[b][n:0]	Master	Read address Where: *n* is equal to address width -1. Address width is in the range of 32-64 bits.
arlen_x[b][3:0]	Master	Burst length for AXI3.
arlen_x[b][7:0]	Master	Burst length for AXI4.
arsize_x[b][2:0]	Master	Burst size.
arburst_x[b][1:0]	Master	Burst type.
arlock_x[b][1:0]	Master	Lock type for AXI3.
arlock_x[b]	Master	Lock type for AXI4.
arcache_x[b][3:0]	Master	Memory type.
arprot_x[b][2:0]	Master	Protection type.
arqos_x[b][3:0]	Master	Quality of Service, QoS. Only when enabled from the GUI.
arregion_x[b][3:0]	Master	Region identifier. Only when enabled from the GUI and from a master interface of the NIC-400.
aruser_x[b][n:0]	Master	User signal. Where: *n* is equal to user defined width -1. A user defined width is in the range of 0-256 bits. If 0 is selected then the signal is not used.
arvalid_x[b]	Master	Read address valid.
arready_x[b]	Slave	Read address ready.

a. You can select uppercase or lowercase signal names from the GUI.

b. Where x is:

<port_name>_s for a bridge slave interface

<port_name>_m for a bridge master interface

<port_name> for a system slave interface or master interface.

A.3.5 Read data channel signals

Table A-7 shows the AXI read data channel signals. Unless the description indicates otherwise, these signals are used by AXI3 and AXI4 protocols. For more information on each signal see the *AMBA AXI and ACE Protocol Specification*.

Table A-7 Read data channel signals

AXI signal NIC-400 adaptation[a]	Source	Description
rid_x[b][n:0]	Slave	Read ID tag. Where *n* is a variable. • For an ASIB**, ID width** {0-16} where 0 indicates that no value has been selected. • For an AMIB, either: — **Global ID width** {1-24} or — if **ID reduction** has been selected, then the values can be viewed from the AMIBs ID reduction report from the GUI.
rdata_x[b][n:0]	Slave	Read data. Where: *n* is equal to data width - 1. Data widths can be 32, 64, 128 or 256 bits.
rresp_x[b][1:0]	Slave	Read response.
rlast_x[b]	Slave	Read last.
ruser_x[b][n:0]	Slave	User defined signal. Where: *n* is equal to user defined width -1. A user defined width is in the range of 0-256 bits. If 0 is selected then the signal is not used.
rvalid_x[b]	Slave	Read valid.
rready_x[b]	Master	Read ready.

a. You can select uppercase or lowercase signal names from the GUI.

b. Where x is:

<port_name>_s for a bridge slave interface

<port_name>_m for a bridge master interface

<port_name> for a system slave interface or master interface.

A.4 APB signals

Table A-8 shows the APB signals. For more information on each signal see the *AMBA APB Protocol Specification*.

Table A-8 APB signals

APB signal NIC-400 adaptation[a]	Source	Description
paddr_x[b]**[31:0]**	APB bridge	Address.
pprot_x[b]**[2:0]**	APB bridge	Address. Only for APB4.
pselx_x[b]	APB bridge	Select.
penable_x[b]	APB bridge	Enable.
pwrite_x[b]	APB bridge	Direction.
pwdata_x[b]**[31:0]**	APB bridge	Write data.
pstrb_x[b]**[3:0]**	APB bridge	Write strobes. Only for APB4.
pready_x[b]	Slave interface	Ready. Only for APB3 or APB4.
prdata_x[b]**[31:0]**	Slave interface	Read data.
pslverr_x[b]	Slave interface	This signal indicates a transfer failure. Only for APB3 or APB4.

a. You can select uppercase or lowercase signal names from the GUI

b. Where:

x is a port name selected by the GUI.

A.5 AHB-Lite signals

The following signals are described:

- *Master and Mirrored master signals*
- *Slave and Mirrored slave signals* on page A-11.

A.5.1 Master and Mirrored master signals

Table A-9 shows the protocol signals generated by a master. For more information on each signal see the *AMBA 3 AHB-Lite Protocol Specification*.

Table A-9 Master signals

AHB-Lite signal NIC-400 adaptation[a]	Destination	Description
haddr_x[b][n:0]	Slave and decoder	System address bus. Where: *n* is equal to address width -1. Address width is in the range of 32-64 bits.
hburst_x[b][2:0]	Slave	The burst type indicates if the transfer is a single transfer or forms part of a burst. Fixed length bursts of 4, 8, and 16 beats are supported.
hmastlock_x[b]	Slave	When HIGH, this signal indicates that the current transfer is part of a locked sequence.
hprot_x[b][3:0]	Slave	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wants to implement some level of protection.
hsize_x[b][2:0]	Slave	Indicates the size of the transfer.
htrans_x[b][1:0]	Slave	Indicates the transfer type of the current transfer.
hwdata_x[b][n:0]	Slave	Write data bus. The write data bus transfers data from the master to the slaves during write operations. Where: *n* is equal to data width -1. Data widths can be 32, 64, 128 or 256 bits.
hrdata_x[b][n:0]	Master	Read data bus. During read operations, the read data bus transfers data from the selected slave to the multiplexor. The multiplexor then transfers the data to the master. Where: *n* is equal to data width -1. Data widths can be 32, 64, 128 or 256 bits.
hwrite_x[b]	Slave	Indicates the transfer direction.
hready_x[b]	Slave	When HIGH, the **hready** signal indicates to the master and all slaves, that the previous transfer is complete.

Table A-9 Master signals (continued)

AHB-Lite signal NIC-400 adaptation[a]	Destination	Description
hresp_x[b]	Master	The transfer response, after passing through the multiplexer, provides the master with additional information on the status of a transfer. • When LOW, the **hresp** signal indicates that the transfer status is OKAY • When HIGH, the **hresp** signal indicates that the transfer status is ERROR.
hwuser_x[b]**[n:0]**	Slave	Write data user defined signal. Where: *n* is equal to user defined width -1. A user defined width in the range of 0-256 bits. If 0 is selected then the signal is not used.
hruser_x[b]**[n:0]**	Master	Read data user defined signal. Where: *n* is equal to user defined width -1. A user defined width in the range of 0-256 bits. If 0 is selected then the signal is not used.
hauser_x[b]**[n:0]**	Slave	Address user defined signal. Where: *n* is equal to user defined width -1. A user defined width in the range of 0-256 bits. If 0 is selected then the signal is not used.

a. Where:
You can select uppercase or lowercase signal names from the GUI.

b. Where x is:
<port_name>_s for a bridge slave interface
<port_name>_m for a bridge master interface
<port_name> for a system slave interface or master interface.

A.5.2 Slave and Mirrored slave signals

Table A-10 shows the protocol signals generated by a slave. For more information on each signal see the *AMBA 3 AHB-Lite Protocol Specification*.

Table A-10 Slave signals

AHB-Lite signal NIC-400 adaptation[a]	Destination	Description
haddr_x[b]**[n:0]**	Slave and decoder	System address bus. Where: *n* is equal to address width -1. Address width is in the range of 32-64 bits.
hburst_x[b]**[2:0]**	Slave	The burst type indicates if the transfer is a single transfer or forms part of a burst. Fixed length bursts of 4, 8, and 16 beats are supported.
hmastlock_x[b]	Slave	When HIGH, this signal indicates that the current transfer is part of a locked sequence.

Table A-10 Slave signals (continued)

AHB-Lite signal NIC-400 adaptation[a]	Destination	Description
hprot_x[b][3:0]	Slave	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wants to implement some level of protection.
hsize_x[b][2:0]	Slave	Indicates the size of the transfer.
htrans_x[b][1:0]	Slave	Indicates the transfer type of the current transfer.
hwdata_x[b][n:0]	Slave	Write data bus. The write data bus transfers data from the master to the slaves during write operations. Where: *n* is equal to data width -1. Data widths can be 32, 64, 128 or 256 bits.
hrdata_x[b][n:0]	Master	Read data bus. During read operations, the read data bus transfers data from the selected slave to the multiplexor. The multiplexor then transfers the data to the master. Where: *n* is equal to data width -1. Data widths can be 32, 64, 128 or 256 bits.
hwrite_x[b]	Slave	Indicates the transfer direction.
hready_x[b]	Slave	When HIGH, the **hready** signal indicates to the master and all slaves, that the previous transfer is complete.
hreadyout_x[b]	Master	When HIGH, the **hreadyout** signal indicates that a transfer has finished on the bus. AHB slave interface only.
hresp_x[b]	Master	The transfer response, after passing through the multiplexer, provides the master with additional information on the status of a transfer. • When LOW, the **hresp** signal indicates that the transfer status is OKAY • When HIGH, the **hresp** signal indicates that the transfer status is ERROR.
hwuser_x[b][n:0]	Slave	Write data user defined signal. Where: *n* is equal to user defined width -1. A user defined width in the range of 0-256. If 0 is selected then the signal is not used.

Table A-10 Slave signals (continued)

AHB-Lite signal NIC-400 adaptation[a]	Destination	Description
hruser_x[b]**[n:0]**	Master	Read data user signal. Where: *n* is equal to user defined width -1. A user defined width in the range of 0-256. If 0 is selected then the signal is not used.
hauser_x[b]**[n:0]**	Slave	Address user defined signal. Where: *n* is equal to user defined width -1. A user defined width in the range of 0-256. If 0 is selected then the signal is not used.
hselx_x[b]	Slave	Slave select signal. AHB slave interface only.

a. You can select uppercase or lowercase signal names from the GUI.

b. Where x is:

<port_name>_s for a bridge slave interface

<port_name>_m for a bridge master interface

<port_name> for a system slave interface or master interface.

A.6 QVN signals

Table A-11 and Table A-12 show the virtual network QVN signals.

Table A-11 VN interface signals

QVN signal NIC-400 adaptation[a]	Source	Description
vawvalid_vn(n)[b]_x[c]	Master	Token request valid. This signal indicates that the master requests a token.
vawready_vn(n)[b]_x[c]	Slave	Token request accepted. This signal indicates that the slave is ready to accept an address and associated control signals.
vawqos_vn(n)[b]_x[c][3:0]	Master	Quality of service value. Non-standard AXI3 signal.
vwvalid_vn(n)[b]_x[c]	Master	Token request valid. This signal indicates that the master requests a token.
vwready_vn(n)[b]_x[c]	Slave	Token request accepted. This signal indicates that the slave is ready to accept a write data transfer and associated control signals.
varvalid_vn(n)[b]_x[c]	Master	Token request valid. This signal indicates that the master requests a token.
varready_vn(n)[b]_x[c]	Slave	Token request accepted. This signal indicates that the slave is ready to accept an address and associated control signals.
varqos_vn(n)[b]_x[c][3:0]	Master	Quality of service value. Non-standard AXI3 signal.

a. You can select uppercase or lowercase signal names from the GUI
b. Where:
n is the virtual network number.
c. Where **x** is:
<port_name>_s for a bridge slave interface
<port_name>_m for a bridge master interface
<port_name> for a system slave interface or master interface.

A.6.1 AXI VN signals

Table A-12 shows the AXI virtual network QVN signals.

Table A-12 AXI VN signals

QVN signal NIC-400 adaptation[a]	Source	Description
awvnet_vn_x[b][3:0]	Master	AW channel virtual network ID
wvnet_vn_x[b][3:0]	Master	W channel virtual network ID
arvnet_vn_x[b][3:0]	Master	AR channel virtual network ID

a. You can select uppercase or lowercase signal names from the GUI.
b. Where b is:
<port_name>_s for a bridge slave interface
<port_name>_m for a bridge master interface
<port_name> for a system slave interface or master interface.

Appendix B
Revisions

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

Table B-1 Issue A

Change	Location	Affects
No changes, first release	-	-