CoreLink™ Network Interconnect NIC-301

Revision: r2p3

Technical Reference Manual

CoreLink Network Interconnect NIC-301
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
05 May 2006	A	Non-Confidential	First release for r0p0.
19 September 2006	B	Confidential	First release for r1p0.
06 July 2007	C	Confidential	First release for r1p1.
02 October 2007	D	Confidential	First release for r1p2.
22 April 2008	E	Confidential	Second release for r1p2. Updates to the following: • Decode register • Read acceptance capability and write acceptance capability • Write issuing capability • Narrow to wide translation • Physical data.
03 November 2009	F	Non-Confidential	First release for r2p0.
19 February 2010	G	Non-Confidential	First issue for r2p1.
03 August 2010	H	Non-Confidential	First issue for r2p2. Changed product name from AMBA® Network Interconnect NIC-301 to CoreLink™ Network Interconnect NIC-301.
29 September 2011	I	Non-Confidential	First issue for r2p3.

Product Status

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

Web Address

`http://www.arm.com`

Contents
CoreLink Network Interconnect NIC-301 Technical Reference Manual

Preface

This preface introduces the *AMBA Network Interconnect (NIC-301)*. It contains the following sections:

- *About this book* on page v
- *Feedback* on page vii.

About this book

This book is for AMBA Network Interconnect.

Product revision status

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

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

Intended audience

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

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level view of the AMBA Network Interconnect and a description of its features.

Chapter 2 *Functional Description*

Read this for a description of the major interfaces and components of the AMBA Network Interconnect. The chapter also describes how they operate.

Chapter 3 *Programmers Model*

Read this for a description the address map and registers of the AMBA Network Interconnect.

Appendix A *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 vi.

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.
`monospace` (mono underlined)	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>`

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:

- *AMBA® Network Interconnect (NIC-301) Integration Manual* (ARM DII 0157)
- *AMBA Network Interconnect (NIC-301) Implementation Guide* (ARM DII 0222)
- *AMBA Designer (ADR-301) User Guide* (ARM DUI 0333)
- *AMBA Network Interconnect (NIC-301) Supplement to AMBA Designer (ADR-301) User Guide* (ARM DSU 0003)
- *AMBA AXI Protocol v1.0 Specification* (ARM IHI 0022)
- *AMBA 3 AHB-Lite Protocol v1.0 Specification* (ARM IHI 0033)
- *AMBA 3 APB Protocol v1.0 Specification* (ARM IHI 0024)
- *APB Rev E Specification* (ARM IHI 0009).

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 0397I
- 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™ Network Interconnect. It contains the following sections:

1.1 About the CoreLink Network Interconnect

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

A CoreLink Network Interconnect configuration can consist of multiple switches with many topology options. Figure 1-1 shows a top-level block diagram of the CoreLink Network Interconnect that contains:

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

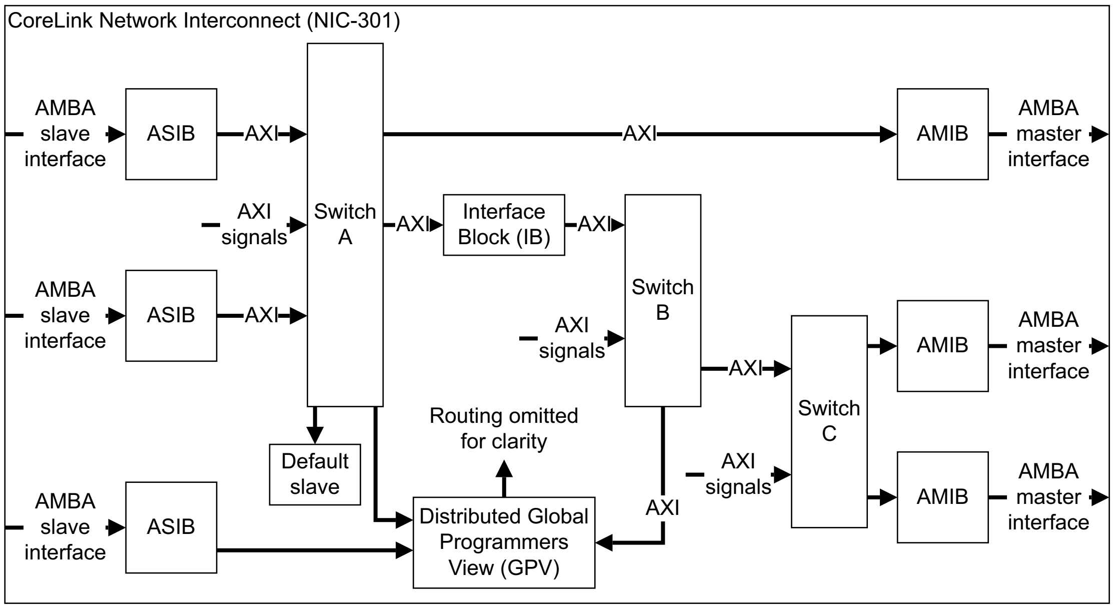

Figure 1-1 CoreLink Network Interconnect top-level block diagram

1.2 Key features

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

- 1-128 AXI or AHB-Lite slave interfaces.
- 1-64 master interfaces that can be AXI, AHB-Lite, APB2, or APB3.
- Configuration of:
  - an APB port to support 1-16 slaves
  - an AXI port to support four region control bits.
- 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 that any master, or a discrete configuration slave interface, can access it. See Chapter 3 *Programmers Model*.
- Highly flexible timing closure options.

1.3 Relationship between CoreLink Network Interconnect and AMBA Designer

AMBA Designer is a configuration tool that generates a specific implementation of a CoreLink Network Interconnect. AMBA Designer drives the CoreLink 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 Network Interconnect and AMBA Designer are designed to be used together to describe the principles of the CoreLink 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 Network Interconnect documentation*
- *AMBA Designer documentation*.

1.3.1 CoreLink Network Interconnect documentation

The CoreLink Network Interconnect documentation consists of:

TRM	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.
IG	The *Implementation Guide* (IG) describes how to set up the network environment and how to use it to run RTL simulations or implementation scripts.
IM	The *Integration Manual* (IM) describes how to integrate a configured network into a larger subsystem.

1.3.2 AMBA Designer documentation

The *AMBA Designer (ADR-301) 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 Network Interconnect (NIC-301) Supplement to AMBA Designer (ADR-301) User Guide* describes how to produce a customized infrastructure.

1.4 Product revisions

This section describes differences in functionality between product revisions of the CoreLink Network Interconnect (NIC-301):

r0p0-r1p0 Contains the following differences in functionality:

- support for 128-bit data width on AXI and all AHB variant interfaces
- support for n-bit addressing on AXI and all AHB variant interfaces, where n is 32-64 bit inclusive
- the decode register is a slave interface property instead of a global property
- single-slave-per-ID *Cyclic Dependency Avoidance Scheme* (CDAS)
- ID register and configuration data
- use of updated synchronous bridges
- use of an AHB to AXI bridge optimized for accessing memory where appropriate.

r1p0-r1p1 Contains the following differences in functionality:

- Separation of arbitration of AW and AR channels so that a slave can accept transactions from two different masters simultaneously.
- Changes to the way that the CoreLink Network Interconnect handles LOCKed transactions.
- Configurable arbitration schemes, that can be:
  - — programmable *Least Recently Granted* (LRG), the scheme present in r1p0
  - — a non-programmable *Round-Robin* (RR) scheme.
- You can select and configure arbitration schemes for each master interface.
- The APB programming interface enables you to program and interrogate the new, separate arbitration schemes.
- The AHB to AXI bridge is optimized for accessing memory is updated with performance enhancements, and to fix a defect.
- The way arbitration schemes are described has changed to enable you to select and configure arbitration schemes.
- The *Quality of Service* (QoS) tidemark value now represents the maximum permitted number of active transactions before the QoS mechanism is activated, instead of the minimum number of unused slots before the mechanism is activated. This means the combined acceptance capability attribute of a master interface is no longer required.

r1p1-r1p2 Contains the following differences in functionality:

- Updates to the example synthesis scripts.
- Addition of a programmable version of the fixed round-robin arbitration scheme.
- Shortening some long paths to improve synthesis performance.
- A change to the way register slices are instantiated. This makes it easier to use them to resolve timing issues during synthesis.
- The choice of CDAS is independent for reads and writes.
- Additional configuration option for the single-slave-per-ID CDAS that permits only a single data-active write transaction to be in progress.

r1p2-r2p0	Contains the following differences in functionality: • Network of interconnects instead of a single interconnect • Optimized translation latency replaces additive translation latency • Single cycle arbitration instead of arbitration switching delay • Enhanced buffering to reduce stalls • Multiple outstanding downsizer transactions instead of limited outstanding downsizer transactions • Upsizer packs data into the wide bus, instead of an expander, for data width changes • Global dynamic QoS instead of local static QoS • Internally programmable instead of externally programmed • System-level address map replaces an address map for each interconnect • Extended timing closure options replace limited timing closure options • 256-bit maximum data width instead of 128-bit maximum data width • Multi-region slave support replaces a single region per slave • Write FIFO, transaction release control instead of fixed write transaction release point. See Appendix A *Revisions*.
r2p0-r2p1	Contains no differences in functionality.
r2p1-r2p2	Contains no differences in functionality.
r2p2-r2p3	Contains no differences in functionality.

Chapter 2
Functional Description

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

2.1 About the functions

You can consider the CoreLink Network Interconnect to be built from functions that each have their 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. For more information, see the *AMBA Designer (ADR-301) User Guide* and CoreLink Network Interconnect (NIC-301) Supplement to AMBA Designer (ADR-301) User Guide.

2.2 Interfaces

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

- *Slave interfaces*
- *Master interfaces* on page 2-8.

2.2.1 Slave interfaces

The CoreLink Network Interconnect supports the following slave interfaces:

- *AXI 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 back to an AHB ERROR.

————————

The AXI DECERR error is mapped back 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.

AXI slave interfaces

An AXI slave interface supports the full AXI protocol.

Configuration options

You can configure the following properties:

- Address width of 32-64 bits.
- Data width of 32, 64, 128, or 256 bits.
- User sideband signal width of 0-32 bits.
- Data width upsize function, see *Upsizing data width function* on page 2-12.
- Data width downsize function, see *Downsizing data width function* on page 2-14.
- 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 **AxPROTx** signal determines the security setting of each transaction, and the slaves that it can access.

- Support for the full AXI protocol.

  Note

  You can achieve a gate count reduction and a performance increase if the attached master does not create any AXI 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 1-32 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 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 2-15.
- Timing isolation:
  - from the external master
  - from the infrastructure.

AHB-Lite slave interfaces

The CoreLink 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 detailed 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 and you configure a GPV port:

`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.

See Chapter 3 *Programmers Model*.

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-like 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-like 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 master mirror interface types.
- Address width of 32-64 bits.
- Data width of 32. 64, 128, or 256 bits.
- Data width upsize function that *Upsizing data width function* on page 2-12 describes.
- Data width downsize function that *Downsizing data width function* on page 2-14 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 SWAP-like 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 sideband signal width of 0-32 bits.

2.2.2 Master interfaces

The CoreLink Network Interconnect supports the following master interfaces:

- *AXI master interfaces*
- *AHB master interfaces* on page 2-9
- *APB master interfaces* on page 2-11.

AXI master interfaces

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

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 upsize function that *Upsizing data width function* on page 2-12 describes.
- User sideband signal width of 0-32 bits.
- Data width downsize function that *Downsizing data width function* on page 2-14 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 master that can access the master interface does not create any AXI lock transactions.

- Write issuing capability of 1-32 transactions.
- Read issuing capability of 1-32 transactions.
- Buffering that *FIFO and clocking function* on page 2-15 describes.
- Timing isolation:
  - from the external slave
  - from the network.

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	INCR	Undefined length
WRAP	2	SINGLE	Two transfers
-	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 Network Interconnect detects it, and a programmable enable bit permits the network to create a DECERR response.

Note

- 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 transaction's size to zeros.

The network breaks any transactions that cross a 1KB boundary into two 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:

- Address width of 32-64 bits.
- Data width of 32, 64, 128, or 256 bits.
- Data width upsize function that *Upsizing data width function* on page 2-12 describes.
- Data width downsize function that *Downsizing data width function* on page 2-14 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 above.
- Support for the full AHB-Lite master protocol.
- Timing isolation:
  - from the external slave
  - from the network.
- User sideband signal width of 0-32 bits.

APB master interfaces

You can configure the APB interface to support a mixture of APB2 or APB3. The APB data width is always 32-bit, and it is therefore never necessary for the APB interface to require the upsize 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 goes ahead. This means that erroneous data bytes might be written to the slave. The masters accessing the APB interface 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 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 downsize function that *Downsizing data width function* on page 2-14 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 that *FIFO and clocking function* on page 2-15 describes
- 1-16 supported APB slaves
- configurable address region sizes
- non-contiguous address regions
- you can configure each APB slave for:
  - APB2 or APB3
  - 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.

2.3 Operation

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

- *Upsizing data width function*
- *Downsizing data width function* on page 2-14
- *FIFO and clocking function* on page 2-15
- *Arbitration* on page 2-17
- *Cyclic Dependency Avoidance Schemes (CDAS)* on page 2-17
- *Lock support* on page 2-18
- *TrustZone technology and security* on page 2-18
- *Remap* on page 2-21.

2.3.1 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 upsize 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 2-13
- *WRAP bursts* on page 2-13
- *Fixed bursts* on page 2-13
- *Bypass merge* on page 2-13
- *Acceptance capability* on page 2-14.

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 2-3 shows how the network converts INCR bursts when it upsizes them.

Table 2-3 Conversion of INCR bursts by the upsize function

INCR burst type	Converted to
64-bit INCR1	Passes through unconverted
64-bit aligned INCR2	INCR1
8-bit aligned INCR8	INCR1, 128-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

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 2-4 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 2-4 Conversion of WRAP bursts by the upsize 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 could 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.

2.3.2 Downsizing data width function

The downsize 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*
- *WRAP bursts*
- *FIXED bursts* on page 2-15
- *Bypass merge* on page 2-15
- *Acceptance capability* on page 2-15.

INCR bursts

The CoreLink 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 2-5 shows how the network converts INCR bursts when it downsizes them.

Table 2-5 Conversion of INCR bursts by the downsize function

INCR burst type	Converted to
Aligned INCR4	INCR8
Unaligned INCR4	INCR7
Aligned INCR9	INCR15 + INCR2

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

The CoreLink Network Interconnect packs INCR bursts with a SIZE smaller than the output data width to match the output width whenever possible, using the upsize transfer function. See *Upsizing data width function* on page 2-12.

WRAP bursts

The CoreLink 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 Network Interconnect converts FIXED bursts to one or more INCR1 or INCRn bursts depending on the downsize ratio. Table 2-6 shows how the network converts FIXED bursts when it downsizes them.

Table 2-6 Conversion of FIXED bursts by the downsize function

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

The CoreLink 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 downsize 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 upsize function. See *Upsizing data width function on page 2-12*.
- 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).

2.3.3 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
- 1:n
- n:1
- ASYNC m:n
- SYNC m:n

  Note

  You can dynamically select this from the ASYNC m:n option if you configure a GPV.

- programmable.

The network automatically determines that the width of the FIFO is the width of the payload. 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 2-7 shows the actions you must take to convert from one mode to another.

Table 2-7 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.
m:n	1:1	Change the clocks, then change the register.
1:1	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 1:n to m:1 by first programming the register to 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*.

2.3.4 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.

2.3.5 Cyclic Dependency Avoidance Schemes (CDAS)

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

Each slave interface of a switch 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 slave interface of a switch:

- 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 to the current destination for that transaction type, the network stalls the transactions until all the outstanding transactions of that type have been completed.

Single slave per ID

This ensures that at a slave interface of a switch:

- 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 slave interface 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. For complex infrastructures, it might be necessary to configure an additional rule for write transactions.

Note

This is only for the single-slave-per-ID CDAS configured slave interfaces.

The extended write rule ensures that if single-slave-per-ID rules are passed, then the network only issues a write transaction to a new destination if all the outstanding write transactions have had the last write data beat transmitted.

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

2.3.6 Lock support

You set support for locked transaction for masters and slaves at configuration time. The CoreLink 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 up-sized or down-sized, it is possible that leading write data are output from the sizing function for the unlocking transaction before all the locked transactions have completed.

2.3.7 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 2-20
- *Internal programmers view* on page 2-20
- *Security checking for master interfaces* on page 2-20.

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 details

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.

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 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.

2.3.8 Remap

Registers in the programmers model control the remap functionality. See Table 3-4 on page 3-8 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 2-1 to Figure 2-5 on page 2-23 show examples of how different remap states interact with each other. Consider a configuration that uses three remap bits. Figure 2-1 shows the memory map when remap is set to 000, representing no remap.

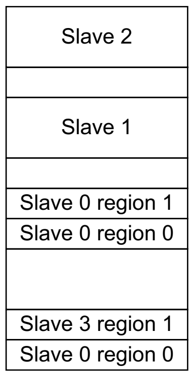

Figure 2-1 No remap, remap set to 000

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 2-2 on page 2-22 shows.

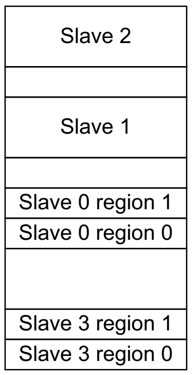

Figure 2-2 Remap set to 001

Alternatively, you can move slave 1 to the bottom of the address range by setting remap to 010 as Figure 2-3 shows.

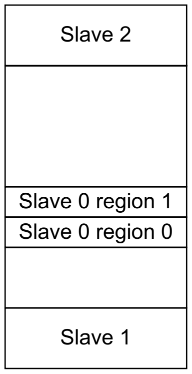

Figure 2-3 Remap set to 010

Note

Remap bit 0 still takes precedence if you set it as Figure 2-4 shows.

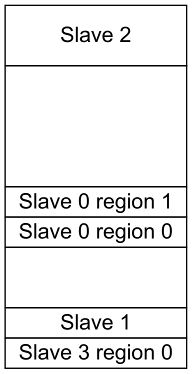

Figure 2-4 Remap set to 011

In addition, you can remove memory regions entirely. Figure 2-5 on page 2-23 shows that if you set remap to 101, Slave 1 is removed.

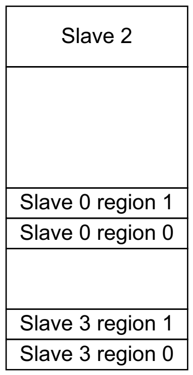

Figure 2-5 Remap set to 101

Chapter 3
Programmers Model

This chapter describes the programmers model.

It contains the following sections:

3.1 About this programmers model

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

3.2 Configuration programmers model

The CoreLink Network Interconnect can contain 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 AMBA Designer (ADR-301).

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 GVP. The GVP:

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

  ——— Note ———

  APROT bit 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) register block. 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 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.

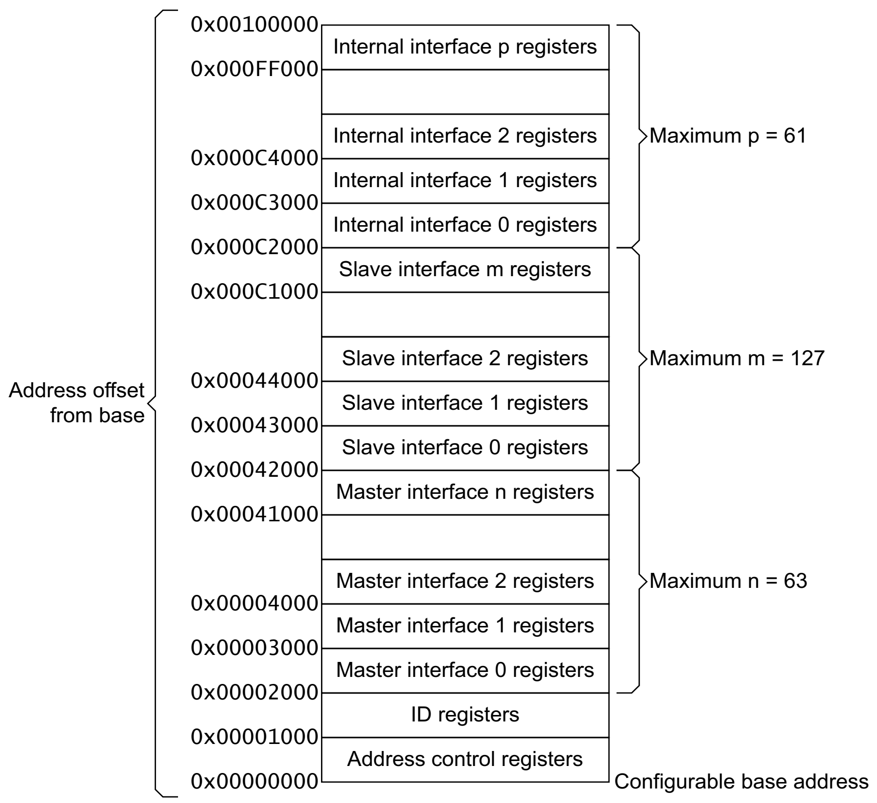
0x00100000
Internal interface p registers
0x000FF000
Internal interface 2 registers
0x000C4000
Internal interface 1 registers
0x000C3000
Internal interface 0 registers
0x000C2000
Slave interface m registers
0x000C1000
Slave interface 2 registers
0x00044000
Slave interface 1 registers
0x00043000
Slave interface 0 registers
0x00042000
Master interface n registers
0x00041000
Master interface 2 registers
0x00004000
Master interface 1 registers
0x00003000
Master interface 0 registers
0x00002000
ID registers
0x00001000
Address control registers
0x00000000
Maximum p = 61
Maximum m = 127
Maximum n = 63
Address offset from base
Configurable base address

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 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 3-8 shows the address region control registers and Table 3-5 on page 3-9 shows the peripheral ID registers.

Note

In Table 3-1 to Table 3-5 on page 3-9, 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 you configure the block as a programmable FIFO. You can configure the register bits as follows: **0** sync 1:1. **1** sync n: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 2-12, *Downsizing data width function* on page 2-14, and *Bypass merge* on page 2-13.
`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, and if not an AHB slave interface. See *FIFO and clocking function* on page 2-15 for information on `wr_tidemark`.
`0x044 - 0x0FC`	-	-	-	-	Reserved.
`0x100`	RW	4	0[b]	`read_qos`	Read channel QoS value.
`0x104`	RW	4	0[b]	`write_qos`	Write channel QoS value.
`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`.
`0x10C - 0xFFC`	-	-	-	-	Reserved.

a. The reset value is initialized to the tidemark value that you set in the configuration GUI in AMBA Designer (ADR-301).

b. If you set the parameter `QoS Type` 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 available if you have a FIFO for all channels. You can configure the register bits for the following clock domain boundaries: **0** sync 1:1. **1** sync n: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 2-12, *Downsizing data width function* on page 2-14, and *Bypass merge* on page 2-13.
`0x028` - `0x03C`	-	-	-	-	Reserved.
`0x040`	RW	4	[a]	`wr_tidemark`	Value, only with a FIFO for the WFIFO channel.
`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
`0x110`	-	-	-	-	Reserved

a. The reset value is initialized to the tidemark value that you set in the configuration GUI in AMBA Designer (ADR-301).

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.
`0x020`	RW	3	4	`Sync_mode`	This register is only available if you have a FIFO for all channels. You can configure the register bits to create different clock domain boundaries as follows: **0** sync 1:1. **1** sync n: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 2-12 and *Downsizing data width function* on page 2-14.
`0x028`	-	-	-	-	Reserved.
`0x040`	RW	4	[a]	`wr_tidemark`	Value, only with a FIFO for the WFIFO channel.
`0x044`	RW	2	-	`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 2-9.
`0x100`	-	-	-	-	Reserved.
`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`.

a. The reset value is initialized to the tidemark value that you set in the configuration GUI in AMBA Designer (ADR-301).

3.2.2 Register blocks

This section contains the following subsections:

- *Address region control*
- *Peripheral ID registers* on page 3-9.

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	-	`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 1, region 5.
`0x0C`	WO	1 - 16	-	`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 1, region 5.
`0x10 - 0x104`	WO	1 - 16	-	`security<n>`	Slave n security setting. It contain 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 1, region 5.
`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
`0xFDC`	RO	8	`0x00`	Peripheral ID7	Reserved
`0xFE0`	RO	8	`0x01`	Peripheral ID0	Part Number [7:0]
`0xFE4`	RO	8	`0xB3`	Peripheral ID1	JEP106[3:0], part number [11:8]
`0xFE8`	RO	8	`0x7B`	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
Revisions

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

Table A-1 Differences between issue E and issue F

Change	Location	Affects
Network of interconnects instead of a single interconnect	Throughout the document	r2p0
Multiple outstanding downsizer transactions instead of limited outstanding downsizer transactions	Throughout the document	r2p0
Upsizer packs data into the wide bus, instead of an expander, for data width changes	Throughout the document	r2p0
Global dynamic QoS instead of local static QoS	Throughout the document	r2p0
Internally programmable instead of externally programmed	Throughout the document	r2p0
System-level address map replaces an address map for each interconnect	Throughout the document	r2p0
Extended timing closure options replace limited timing closure options	Throughout the document	r2p0
256-bit maximum data width instead of 128-bit maximum data width	Throughout the document	r2p0
Multi-region slave support replaces a single region per slave	Throughout the document	r2p0
Write FIFO, transaction release control instead of fixed write transaction release point	Throughout the document	r2p0

Table A-2 Differences between issue F and issue G

Change	Location	Affects
No technical changes	-	-

Table A-3 Differences between issue G and issue H

Change	Location	Affects
Added information to write acceptance capability and read acceptance capability bullet points	*AXI slave interfaces* on page 2-3	r2p2
Added description about transactions received without all four WSTRBs asserted or negated	*AHB master interfaces* on page 2-9	r2p2
Changed 64-bit unaligned WRAP bursts to 128-bit unaligned WRAP bursts to	Table 2-4 on page 2-13	r2p2
Rewrote section about all clock boundary crossings being implemented using a FIFO structure with appropriate synchronization for the current mode of operation	*FIFO and clocking function* on page 2-15	r2p2
Added text to explain that the base address of a register block is determined by the node number assigned to it and that here is no requirement for register blocks to be contiguous after Figure 3-1 on page 3-4	*Register block types* on page 3-3	r2p2
Replaced the term 'quality value' with the term 'QoS value'	Throughout the document	r2p2
Replaced the register name '`fn_mod_iss`' with the name '`fn_mod`'	• Table 3-1 on page 3-4 • Table 3-2 on page 3-6 • Table 3-3 on page 3-7	r2p2
Updated the description for the `wr_tidemark` register	Table 3-3 on page 3-7	r2p2
Modified address offsets for `security0`, `security1`, and `security<n>` registers	Table 3-4 on page 3-8	r2p2
Removed note on page 3-9	*Peripheral ID registers* on page 3-9	r2p2
Added a note to clarify accesses to the GPV	*Configuration programmers model* on page 3-3	r2p2
Updated the reset value entry of `read_qos` and `write_qos`	Table 3-1 on page 3-4	r2p2
Updated the reset value entry of register Peripheral ID 2	Table 3-5 on page 3-9	r2p2

Table A-4 Differences between issue H and issue I

Change	Location	Affects
Added *User sideband signal width of 0-32 bits* to bullets	*Configuration options* on page 2-7 *Configuration options* on page 2-10	All revisions
Second column title changed from *AxLen* to *Number of transfers in AXI transaction* to reflect accuracy of information	Table 2-2 on page 2-9	All revisions
Note underneath table updated for clarification	Table 2-2 on page 2-9	All revisions
Note added to WRAP bursts	*WRAP bursts* on page 2-14	All revisions
Note beneath table changed from (1:n to 1:m) to (1:n to m:1)	Table 2-7 on page 2-16	All revisions

Table A-4 Differences between issue H and issue I (continued)

Change	Location	Affects
Added information on restrictions for GPV	*Configuration programmers model* on page 3-3	All revisions
Description enhancement to Address offset range `0x10` - `0x104`	Table 3-4 on page 3-8	All revisions
Peripheral ID2 Reset value changed from `0x6B` to `0x7B`	Table 3-5 on page 3-9	r2p3