RapidIO™ Interconnect Specification Part 13: AMBA-RapidIO Scale-Out Logical Specification

4.0, 03/2016

RapidIO.org

Revision History

Revision	Description	Date
0.1	First draft for review.	03/07/2016
0.2	Second draft for review.	03/11/2016

Table of Contents

Table of Contents

List of Figures

List of Figures

Blank page

List of Tables

List of Tables

Blank page

Chapter 1 Overview

1.1 Introduction

This chapter provides an overview of the *Part 13: AMBA-RapidIO Scale-Out Logical Specification*, including a description of the relationship between this part and other parts of the RapidIO specification.

1.2 Overview

The RapidIO interconnect was designed to allow both the number of nodes in a system, and the size of each node, to scale. The number of nodes can be scaled using messaging and non-cache-coherent read/write/atomic transactions. The *Part 5 Globally Shared Memory Logical Specification*, which supports a cache coherent non-uniform memory access (CC-NUMA) programming model over a RapidIO fabric, allows multiple devices to behave as a single much larger node. Part 5 is not optimized for any particular processor architecture or internal bus architecture.

RapidIO was chosen as the standard interconnect technology for cache coherency between ARM devices. Cache-coherency is well supported in single-node, multi-core ARM based systems using the AMBA coherency specification[1]. The *Part 13: AMBA-RapidIO Scale-Out Logical Specification* was defined to optimize performance for AMBA ACE coherency across a RapidIO fabric.

The logical specifications do not imply a specific transport or physical interface, therefore they are specified in a bit stream format. Necessary bits are added to the logical encodings for the transport and physical layers that are lower in the specification hierarchy.

RapidIO is a definition of a system interconnect. System concepts such as processor programming models, memory coherency models, and caching are beyond the scope of the RapidIO architecture. The support of memory coherency models, through caches, memory directories (or equivalent, to hold state and speed up remote memory access) is the responsibility of endpoints (processors, memory, and possibly I/O devices), using RapidIO operations.

This document assumes the user of this specification is familiar with the cache coherency protocol defined in the AMBA coherency Specification[1].

1. *AMBA4 ACE Specification* (Issue E, Feb 22, 2013), ARM Inc.

1.2.1 Memory System

Under the globally shared distributed memory programming model, memory may be physically located in different places in the machine yet may be shared among different processing elements. Typically, mainstream system architectures have addressed shared memory using transaction broadcasts sometimes known as bus-based snoopy protocols. These are usually implemented through a centralized memory controller that all devices have equal or uniform access.

Super computers, massively parallel, and clustered machines that have distributed memory systems must use a different technique from broadcasting for maintaining memory coherency. Because a broadcast snoopy protocol in these machines is not efficient – due to the number of devices that must participate and the latency and transaction overhead involved – coherency mechanisms such as memory directories or distributed linked lists are required to keep track of where the most current copy of data resides. These schemes are often referred to as cache coherent, non-uniform memory access (CC-NUMA) protocols. A typical distributed memory system architecture is shown in Figure 1-1.

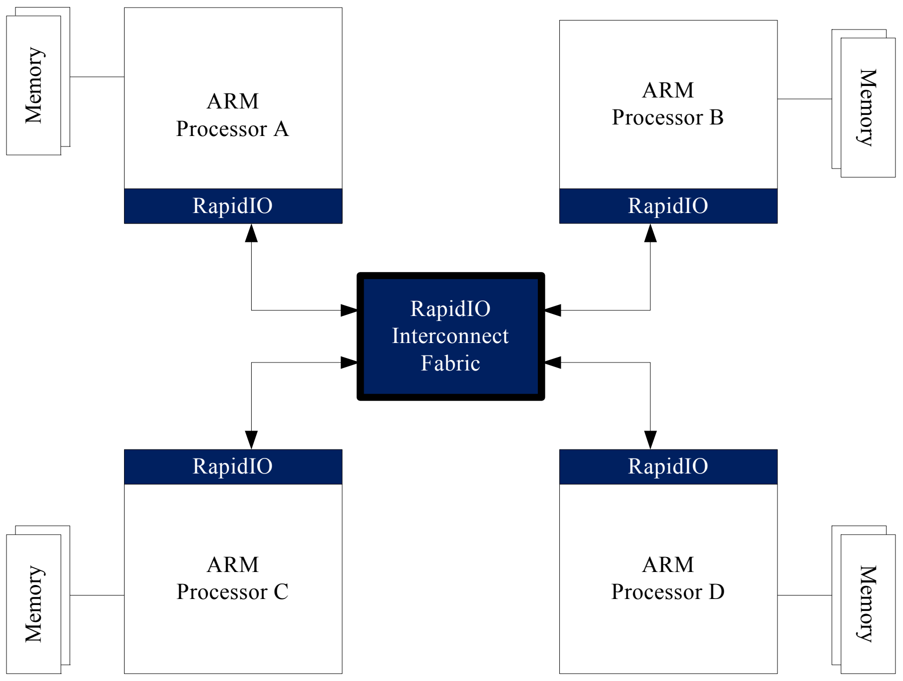


Figure 1-1. Distributed Memory System

For AMBA coherency extension over RapidIO, a flexible approach is chosen where the user may use either a directory or a snoop-filter based scheme to implement the coherency protocol. In this specification, for illustration purposes, a directory based implementation is assumed.

1.3 Features of the AMBA-RapidIO Scale-out Coherency Specification

The following are features of the *Part 13: AMBA-RapidIO Scale-Out Logical Specification*.

1.3.1 Functional Features

- A cache coherent, non-uniform memory access (CC-NUMA) system architecture is enabled to provide a globally shared memory model.
- Interworking with ARM’s AMBA cache coherency bus technology.
- The size of processor memory requests are either in the cache coherence granularity, or smaller. The coherence granule size may be different for different processor families or implementations. This specification supports up to 256 bytes of payload.

1.3.2 Physical Features

- The protocols and packet formats are independent of the physical interconnect and network topology. The protocols work whether the physical fabric is a point-to-point ring, a torus, a switched multi-dimensional network, etc.
- RapidIO is not dependent on the bandwidth or latency of the physical fabric.
- The protocols handle out-of-order packet transmission and reception.
- Certain devices have bandwidth and latency requirements for proper operation. RapidIO does not preclude an implementation from imposing these constraints within the system.

1.3.3 Performance Features

- Packet headers must be as small as possible to minimize the control overhead and be organized for fast, efficient assembly and disassembly.
- 64-bit addresses shall be supported.
- Multiple transactions may be processed in parallel to ensure high throughput.

1.3.4 Scale-out Specification Summary

Table 1-1 provides a comparison summary and an overview of the specification outlined in this document.

fields

Table 1-1. ARM-RapidIO Scale-out Specification Summary

Features	ARM AMBA ACE Specification	RapidIO GSM Specification	ARM-RapidIO Specification
Specification version	*AMBA4 ACE and ACE-Lite* (Issue E, 22 Feb, 2013)	*RapidIO GSM Logical Specification*, (Version 3.0, Oct. 2013)	*Part 13: AMBA-RapidIO Scale-Out Logical Specification*
Architecture	CC-NUMA	CC-NUMA	CC-NUMA
Coherency State Model	5-state Coherency model	3-state coherency model	Supports 5-state coherency as defined in the AMBA ACE specification
Cache-line and Payload Size	Cache-line and Large buffer Granularity Others possible 64-byte (baseline)	Cache-line granularity 64-byte (baseline)	Cache-line and Large buffer granularity. Supports Payload (Coherence Granule) up-to 256 byte
Implementation Aspect	Various options: Snoop Snoop-Filter Directory	Directory	Snoop Snoop-Filter Directory
Endianness	Little Endian or Big Endian (Byte invariant Big Endian)	Big Endian, 64-bit aligned	Big Endian, 64-bit aligned
Ordering, QoS, and Deadlock	AxID for ordering AxQoS for QoS There are independent channels for Request, Snoop, and response for Deadlock avoidance	srcTID for transaction ID Ordering and priority rules for deadlock free system	Priority fields (prio and crf) for deadlock avoidance. New fields for QoS guarantee. Ordering by protocol design.
Packet Format	AMBA ACE bus specific transaction, channels, and signals	Part 5: GSM Packet Format for Request/Response	Various packet format to transport AMBA ACE transactions over RapidIO

1.4 Contents

The *Part 13: AMBA-RapidIO Scale-Out Logical Specification* consists of the following:

- Chapter 1, “Overview,” provides an overview of the ARM-RapidIO scale-out cache coherency specification.
- Chapter 2, “System Models,” introduces heterogeneous system model and supported state models.
- Chapter 3, “Operation Descriptions,” describes the supported transactions, priority assignment schemes to ensure deadlock free communication, and a set of example operations.
- Chapter 4, “Packet Format Descriptions,” contains the packet format definitions for the specification.

- Chapter 5, “Scale-out Coherency Registers,” describes the visible register set that allows an external processing element to determine the scale-out specification capabilities, configuration, and status of a processing element using this logical specification. Only registers or register bits specific to the logical specification are explained. Refer to other RapidIO logical, transport, and physical specifications to determine a complete list of registers and bit definitions.

1.5 Terminology

Refer to the “Glossary of Terms and Abbreviations” on page 45” in this document.

1.6 Conventions

||	Concatenation, used to indicate that two fields are physically associated as consecutive bits
ACTIVE_HIGH	Names of active high signals are shown in uppercase text with no overbar. Active-high signals are asserted when high and not asserted when low.
$\overline{\text{ACTIVE\_LOW}}$	Names of active low signals are shown in uppercase text with an overbar. Active low signals are asserted when low and not asserted when high.
italics	Book titles in text are set in italics.
REG[FIELD]	Abbreviations or acronyms for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets.
TRANSACTION	Transaction types are expressed in all caps.
operation	Device operation types are expressed in plain text.
n	A decimal value.
[*n*-*m*]	Used to express a numerical range from *n* to *m.*
0b*nn*	A binary value, the number of bits is determined by the number of digits.
0x*nn*	A hexadecimal value, the number of bits is determined by the number of digits or from the surrounding context (for example, 0x*nn* may be a 5-, 6-, 7-, or 8-bit value).
x	This value is a don’t care

Chapter 2 System Models

2.1 ARM Processor based Heterogeneous System Model

A variety of system architectures can be developed based on the ARM-RapidIO Scale-out specification. Figure 2-1 shows a heterogeneous disaggregated system example based on Processor, DSP, GPU, FPGA, storage/memory modules and RapidIO fabric.

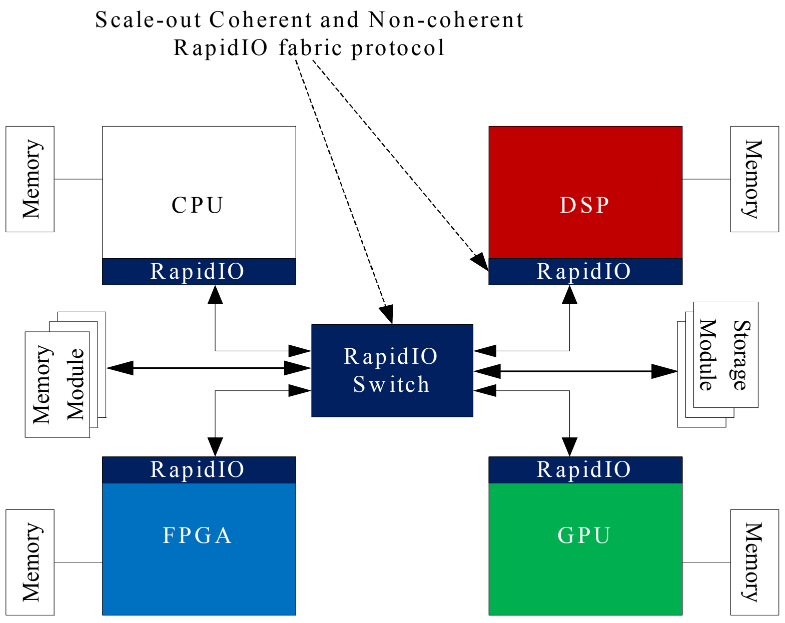


Figure 2-1. Heterogeneous Scale-out System based on ARM Processor and RapidIO Fabric

Figure 2-2 shows an example of a multi-core ARM SoC with on-chip cache-coherent interconnect and RapidIO integrated endpoint. This example supports both cache-coherent and non-coherent scale-out across multiple processing elements in a cluster.

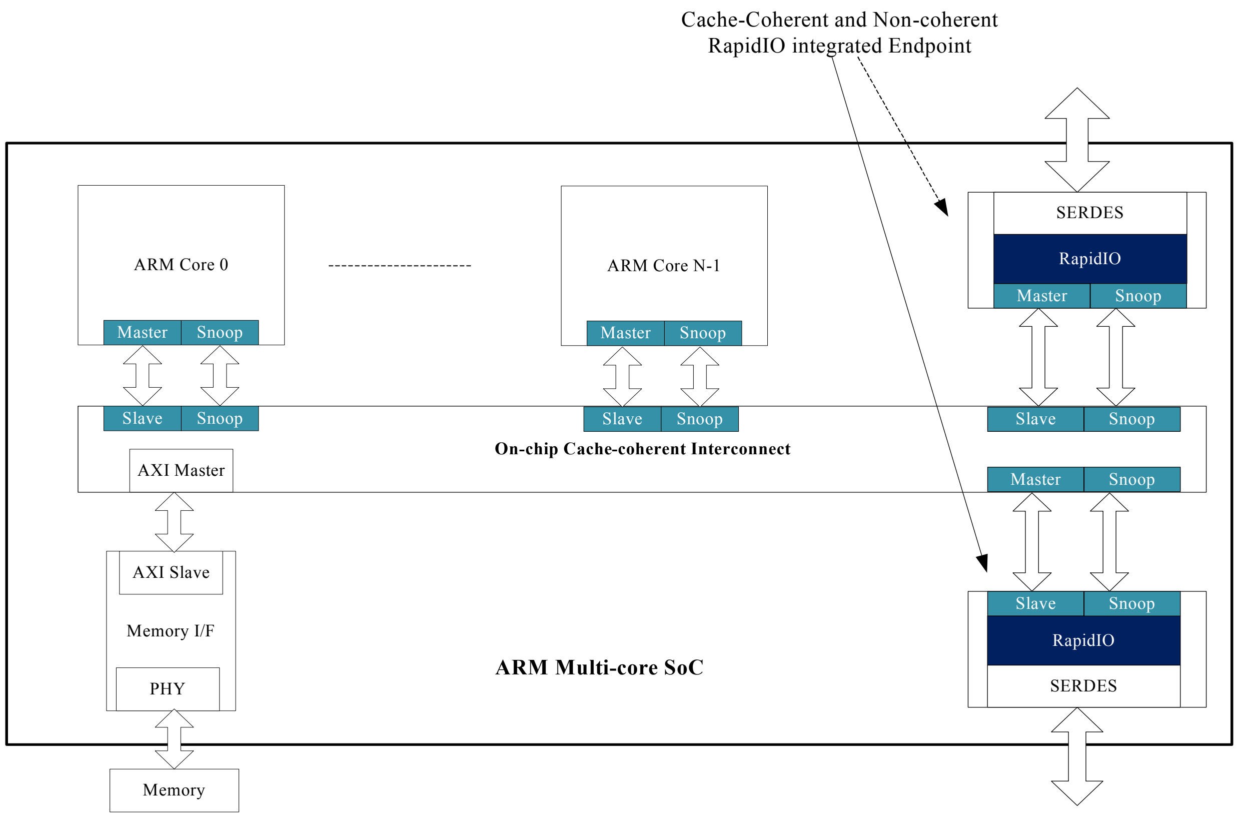


Figure 2-2. ARM Multi-core SoC with Coherent and Non-coherent Scale-out Using RapidIO

2.2 Programming Model

The preferred programming model for modern multiprocessor computer systems provides memory that is accessible from all processors in a cache coherent fashion. This model is also known as the globally shared memory model. Within a single device, bus architectures such as AMBA ACE allow all participants in the cache coherence mechanism to monitor all memory activity simultaneously.

This specification assumes that the memory in the system, which may be distributed across multiple devices, is viewed as a single large global address space. With a distributed memory system, cache coherence is maintained by tracking memory access activity and notifying specific cache coherence participants when a cache coherence hazard is detected. For example, if a processing element wishes to write to a memory address, all participant processing elements that have accessed that coherence granule are notified to invalidate that address in their caches. Only when all of the participant processing elements have completed the invalidate operation and replied back to the tracking mechanism is the write allowed to proceed.

As an example, the tracking mechanism for this specification could rely on the memory directory based system model. This system model allows efficient, moderate scalability with a reasonable amount of information storage required for the tracking mechanism. For example, in an example implementation, a dual directory scheme could be used, where the first directory keeps track of the on-chip coherency, and the second directory – located in the integrated RapidIO endpoint – keeps track of the system wide coherency.

2.3 Coherency State Model

The scale-out specification discussed in this document supports various AMBA ACE protocol specific cache line characteristics and cache line models. In particular, the AMBA ACE protocol extends the AXI4 protocol and supports a five-state cache model to define the state of any cache line in the coherent system. Table 2-1 summarizes the supported cache line characteristics and Figure 2-3 summarizes the supported cache state model. For more information, see the AMBA ACE specification.

Table 2-1. AMBA ACE Cache Line Characteristics

Cache Line Characteristics	Description
Valid, Invalid	When valid, the cache line is present in the cache.
Unique, Shared	When unique, the cache line exists only in one cache. When shared, the cache line might exist in more than one cache but this is not guaranteed.
Clean, Dirty	When clean, the cache does not have responsibility for updating main memory. When dirty, the cache line has been modified with respect to main memory, this cache must ensure that main memory is eventually updated.

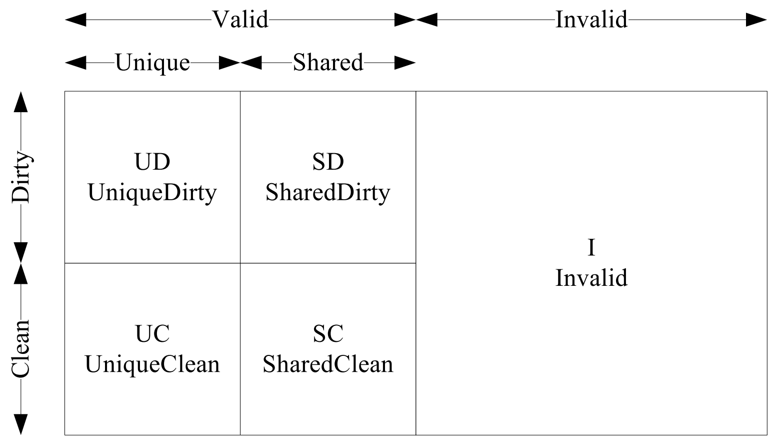


Figure 2-3. ACE Cache State Model Supported by RapidIO

Chapter 3 Operation Descriptions

3.1 Introduction

This chapter defines the interworking protocol between AMBA and RapidIO.

3.2 Supported AMBA Transactions

The following table lists the AMBA transactions supported by this specification.

Table 3-1. Supported AMBA ACE Transactions

Transaction Group	Transaction Type
Non-snooping	ReadNoSnoop
	WriteNoSnoop
Coherent	ReadOnce
	WriteUnique
	WriteLineUnique
	ReadShared
	ReadClean
	ReadNotSharedDirty
	ReadUnique
	CleanUnique
	MakeUnique
CacheMaintenance	CleanShared
	CleanInvalid
	MakeInvalid
DVM	DVMComplete
	DVMMessage
MemoryUpdate	Evict
	WriteBack
	WriteClean

Table 3-1. Supported AMBA ACE Transactions (Continued)

Transaction Group	Transaction Type
Snoop	SnoopReadOnce
	SnoopReadClean
	SnoopReadShared
	SnoopReadNotSharedDirty
	SnoopReadUnique
	SnoopCleanInvalid
	SnoopMakeInvalid
	SnoopCleanShared

3.3 Operation Examples

3.3.1 Read Operation

Figure 3-1 to Figure 3-3 show an example of a Read operation, where Core 0 from SoC1 is performing a read using a ReadShared transaction and can accept a cache line in any state.

The on-chip cache-coherent interconnect (CCI) in SoC1 decodes the address and determines if an off-chip read is required to get the cache line and forward the transaction to the RapidIO endpoint through the CCI master interface over the Read Address channel.

The RapidIO endpoint in SoC1 inspects its memory map and identifies that SoC0 is the home memory for the corresponding cache line. The endpoint controller in SoC1 translates various fields from the AMBA ACE transaction to the RapidIO packet format and sends a read request transaction to SoC0.

The RapidIO endpoint in SoC0 keeps track of the state of its cache-lines and identifies that the requested data from SoC1 exists in the memory as shared. The endpoint in SoC0 generates a ReadShared transaction through the master interface over the Read Address Channel towards the CCI in SoC0.

Once the requested data is received from the CCI, the RapidIO endpoint in SoC0 generates a response towards SoC1 and adds SoC1 in its sharing mask.

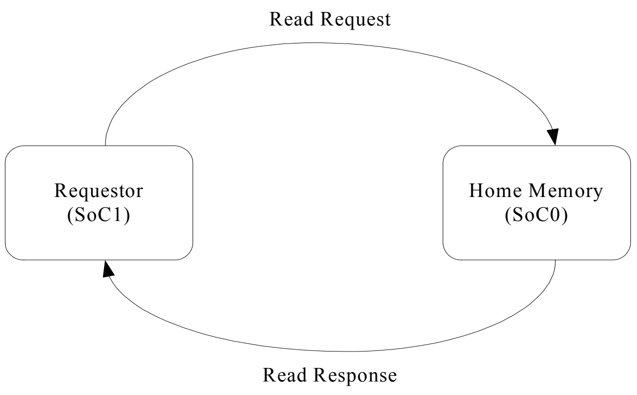


Figure 3-1. Read Operation Example - Top Level

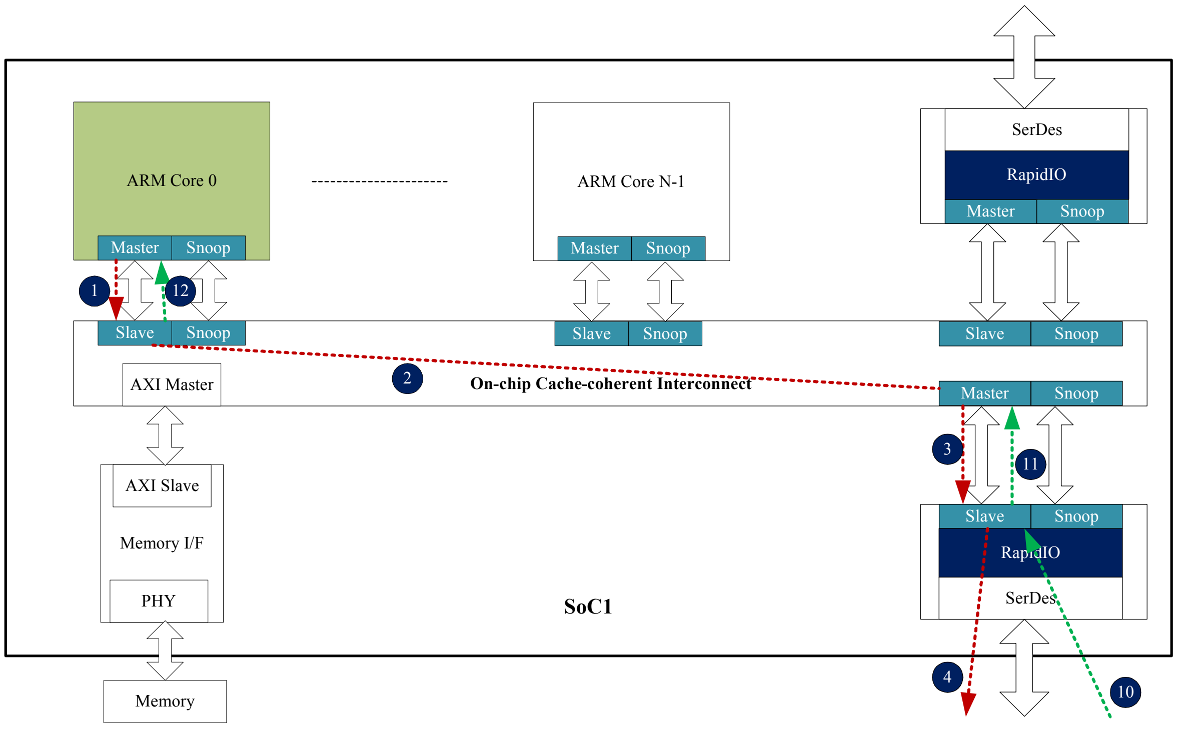


Figure 3-2. Read Operation Example - Requestor (SoC1)

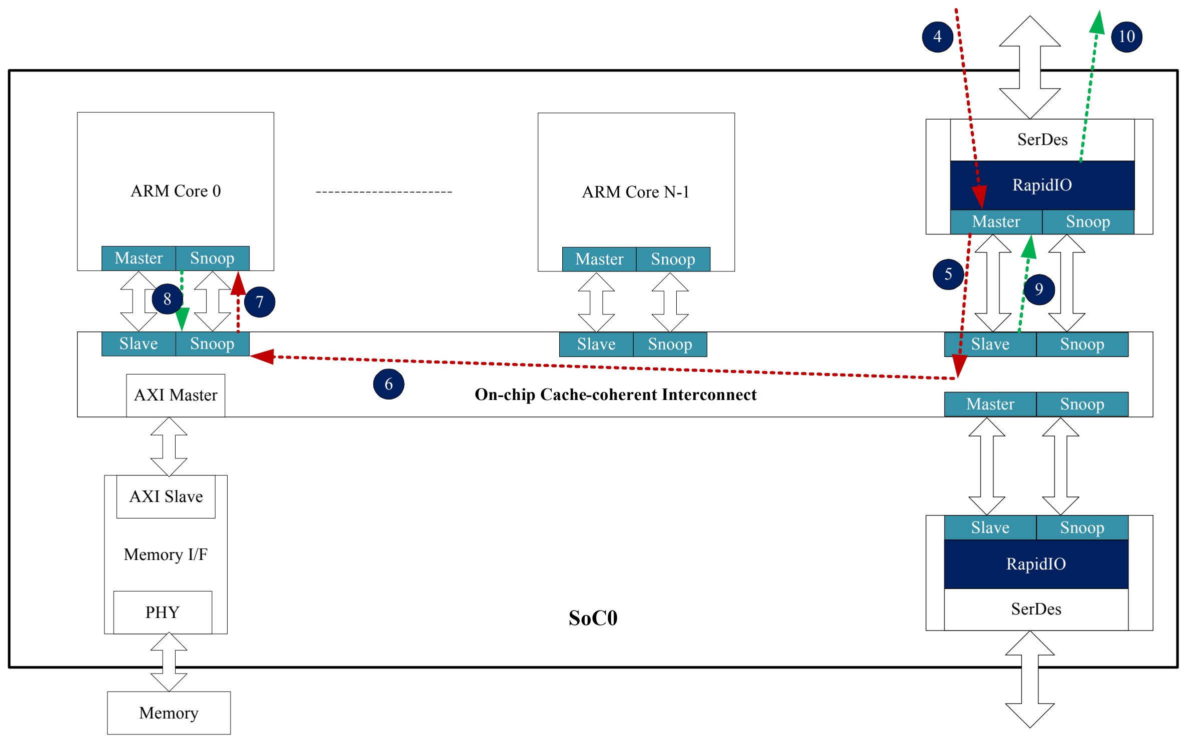


Figure 3-3. Read Operation Example - Responder (SoC0, Home)

3.3.2 ReadUnique Operation

This operation is performed by a processing element to obtain line in a Unique state, i.e., the processing element wants a write permission on a cache line. A set of transactions is used between the participating processing elements to complete the read to own operation. Figure 3-4 shows the message sequence chart for this operation.

In this example, a multi-core 4-node system is assumed, where Core0 in SoC1 wants to have the write permission on the cache line, SoC0 is the home memory for the corresponding cache line, and SoC2 and SoC3 are the sharers.

The transaction types (TTYPE-H/-T) shown in the diagram are discussed in Table 4-6.

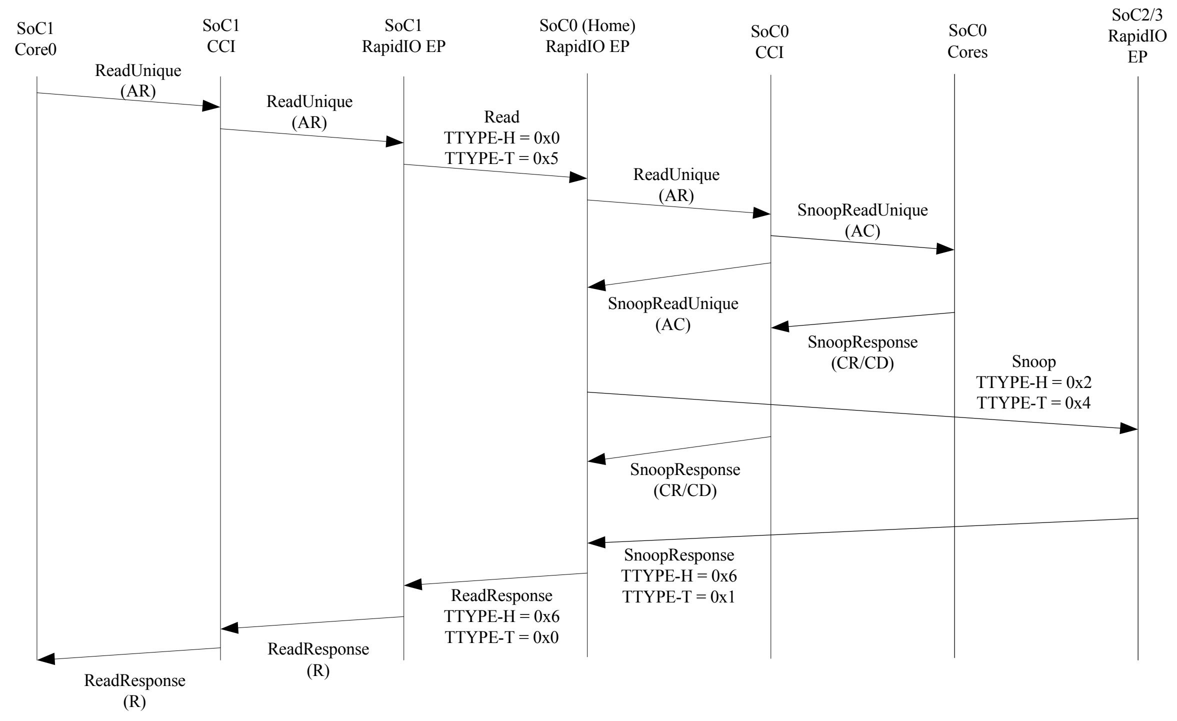


Figure 3-4. Read-to-Own Operation Message Sequence Chart

The message sequence chart is summarized as follows:

1. SoC1 Core0 master sends ReadUnique transaction on the Read Address channel (AR) to the on-chip interconnect.
2. The on-chip interconnect decodes the address and determines that the Home is on another chip (i.e. SoC0).
3. The ReadUnique transaction is sent to the RapidIO interface on the Read Address channel.
4. The RapidIO endpoint decodes the address to determine which chip is the Home for the address. The endpoint IP maps the AMBA ACE transactions, forms the packet and adds the correct Target ID, then sends across the off-chip interface.
5. The ReadUnique transaction request packet arrives at the off-chip interface.
6. The transaction is translated to ACE and sent as a ReadUnique transaction, on the Read Address channel (AR), to the on-chip interconnect.
7. The on-chip interconnect decodes the address and determines that it is the Home for the address.
8. The Home looks up in a snoop filter to determine which components might have a copy of the address.
   a) For the on-chip sharers, it sends a SnoopReadUnique snoop to each on the Snoop Address channel (AC).
   b) For the off-chip sharers, it sends a single SnoopReadUnique snoop to the RapidIO interface on the Snoop Address channel (AC).

9. The RapidIO endpoint interface will receive a single SnoopReadUnique snoop and it will then look-up in its own directory to determine which other chips might have a copy of the line. It sends a separate snoop to each other chip that might have a copy.
10. When the snoop transactions arrives at the RapidIO endpoint of another chip (e.g., SoC2 and SoC3)
    a) It looks up in its directory (note this is different from the directory in step 9) to determine which cores on its chip might have a copy of the line and it sends a snoop to each of them on the Snoop Address channel (AC).
    b) It waits for a snoop response on the Snoop Response channel (CR), and possibly on the Snoop Data Channel (CD), from every CPU that it has sent a snoop to.
    c) It collates all the responses and sends a single snoop response back to the SoC with the Home node.
11. The snoop response arrives back at SoC0. It may or may not have data associated with it.
12. The Home, on SoC0, collates all the snoop responses on the Snoop Response channel (CR) and possibly on the Snoop Data channel (CD). This should show that all copies have been invalidated.
13. The Home obtains the data for the location, either from one of the snoop responses or from main memory.
14. The Home send the ReadUnique data and response on the Read Data channel (R) back to the RapidIO interface.
15. The RapidIO interface on SoC0:
    a) Sends the ReadUnique data and response back to SoC1
    b) Records in its directory that SoC1 has a copy of the line
16. The ReadUnique data and response arrive back at the RapidIO endpoint interface on SoC1, which then:
    a) Sends the ReadUnique data and response back to CPU0 on the Read Data channel (R)
    b) Records in its directory that Core0 has a copy of the line
17. CPU0 in SoC1 receives the read data and response for the transaction on the Read Data channel (R).

3.4 AMBA ACE to RapidIO Interworking

Table 3-2. Example Interworking Table

SoC1 CCI	**SoC1 RapidIO Transaction**	**Home Memory RapidIO Response**	**Home Memory Cache Line State**	**Home Memory RapidIO Request to SoC 0**	**SoC 0 CCI Internal**	**SoC 0 RapidIO Response**	**Comments**
ReadNoSnoop	ReadNoSnoop	Response with Data	N/A	N/A	N/A	N/A	ReadNoSnoop must access memory that is not shared.
WriteNoSnoop	WriteNoSnoop	Response no Data	N/A	N/A	N/A	N/A	WriteNoSnoop must access memory that is not shared.
ReadOnce	ReadOnce	Response with Data	N/A	N/A	N/A	N/A	Only need an copy of the value, not the latest.
WriteUnique	WriteUnique	Response no Data	I, UC, SC	Write Unique	Write Unique	Response No Data	Cache line written must not be dirty.
			UD, SD	Clean Unique	Clean Unique	Response with Data	If the line to be written is dirty, the dirty cache lines must be pushed to home memory before being updated by the WriteUniqe. Note special cases of overlapping WriteUnique in AMBA ACE specification.
WriteLineUniq ue	WriteLineUnique	Response no Data	Invalid	N/A	N/A	N/A	Nobody to update
			UC, SC, UD, SD	MakeInvali d	MakeInvali d	Response without Data	If the entire cache line is written, all other copies become invalid.
ReadShared	ReadShared	Response with Data	Any	N/A	N/A	N/A	ReadShared accepts a cache line in any state.
ReadClean	ReadClean	ReadClean	I, UC, SC	N/A	N/A	N/A	Read clean requires the cache line to be clean.
			UD, SD	CleanUniq ue	CleanUniq ue	Response with Data	The cache line must be forced back to the home memory before being passed back to the originator.

3.5 System Issues

3.5.1 Transaction Delivery

Cache coherent packets transmitted by a processing element shall be delivered in order by the RapidIO interconnect.

Switch devices which support *Part 13: AMBA-RapidIO Scale-Out Logical Specification* shall maintain the order of multicast and non-multicast cache coherent packets.

Cache coherent transactions are not ordered with respect to flows of other transaction types. Implementations shall not assume any ordering relationship between cache coherency transactions and other RapidIO logical layer transactions.

3.5.2 Deadlock Considerations and QoS

The method by which a RapidIO system maintains a deadlock free environment is described in *RapidIO Interconnect Specification Part 6: LP-Serial Physical Layer Specification*. The AMBA ACE protocol imposes the following additional requirements to avoid deadlock:

- Response transactions must make forward progress regardless of Snoop and Request transactions.
- Snoop transactions must make forward progress regardless of Request transactions.

To ensure these rules are maintained, Table 3-3 summarizes the physical layer priority relationships that shall be used for the three types of AMBA ACE transactions (Response, Snoop, and Request) in the system.

Table 3-3. Priority Assignment for AMBA ACE Transactions in RapidIO

Transactions	Priority Level	Example Physical Layer Packet Priority Value
Response, Snoop Response	Highest	2’b11
Snoop	Middle	2’b01
Request	Lowest	2’b00

The AMBA ACE bus protocol defines a 4 bit QoS signal associated with each bus transaction. Devices which support the *Part 13: AMBA-RapidIO Scale-Out Logical Specification* shall transport the 4-bit AMBA QoS field (AxQoS) using the 4-bit axQoS field defined in this specification. See Section 4.4, “Field Mapping” for more information.

In addition, devices that support this specification shall determine the value of the RapidIO physical layer Virtual Channel (VC), and Critical Request Flow (CRF) bits for request and snoop packets based on the value of the QoS signals.

The specific physical priority levels used for RapidIO packets may vary depending on the capabilities of the fabric and topology.

Performance may be optimized by setting the priority of snoop responses to be different from that of responses. For these reasons, RapidIO endpoints which support the *Part 13 ARM-RapidIO Scale-Out Cache-Coherency Logical Specification* shall allow the value of the Virtual Channel (VC), Critical Request Flow (CRF) and priority (prio) physical layer bits of the request, snoop, snoop response, and response packets to be independently programmable for each QoS bus value supported by the device and to guarantee deadlock avoidance.

The priority of response and snoop response packets shall be determined by the QoS value of the received request and snoop request packet.

Table 3-4 shows an example mapping from AMBA ACE QoS value to RapidIO physical layer VC/CRF/priority values.

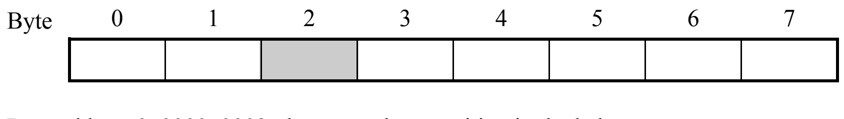
Table 3-4. Example Mapping QoS to VC/CRF/prio

Flow	QoS Value	Packet Type	VC	CRF	prio
High Priority Flow	4’b1010	Snoop Response	1’b0	1’b1	2’b11
		Response	1’b0	1’b1	2’b11
		Snoop	1’b0	1’b1	2’b01
		Request	1’b0	1’b1	2’b00
Low Priority Flow	4’b0000	Snoop Response	1’b0	1’b0	2’b11
		Response	1’b0	1’b0	2’b10
		Snoop	1’b0	1’b0	2’b01
		Request	1’b0	1’b0	2’b00

3.5.3 Endian, Byte Ordering, and Alignment

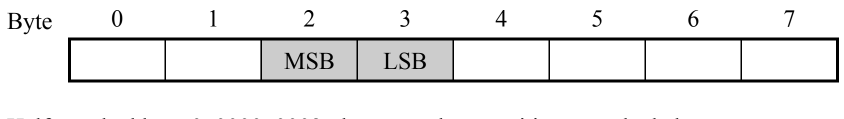
RapidIO has double-word (8-byte) aligned big-endian data payloads. This means that ARM based devices with little-endian format shall perform the proper endian conversion to format transactions.

Operations that specify data quantities that are less than 8 bytes shall have the bytes aligned to their proper byte position within the big-endian double-word, as shown in Figure 3-5 to Figure 3-7.



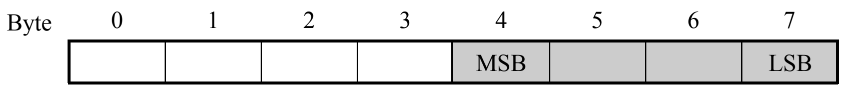
Byte address 0x0000_0002, the proper byte position is shaded.

Figure 3-5. Byte Alignment Example



Half-word address 0x0000_0002, the proper byte positions are shaded.

Figure 3-6. Half-Word Alignment Example



Word address 0x0000_0004, the proper byte positions are shaded.

Figure 3-7. Word Alignment Example

For write operations, a processing element shall properly align data transfers to a double-word boundary for transmission to the destination. This alignment may require breaking up a data stream into multiple transactions if the data is not naturally aligned. A number of data payload sizes and double-word alignments are defined to minimize this burden.

Figure 3-8 shows a 48-byte data stream that a processing element wishes to write to another processing element through the interconnect fabric. The data displayed in the figure is big-endian and double-word aligned with the bytes to be written shaded in grey. Because the start and end of the stream are not aligned to a double-word boundary, the sending processing element shall break the stream into three transactions as shown in the figure.

The first transaction sends the first 3 bytes (in byte lanes 5, 6, and 7) and indicates a byte lane 5, 6, and 7 three-byte write. The second transaction sends all of the remaining data except for the final sub double-word. The third transaction sends the final 5 bytes in byte lanes 0, 1, 2, 3, and 4 indicating a 5-byte write in byte lanes 0, 1, 2, 3, and 4.

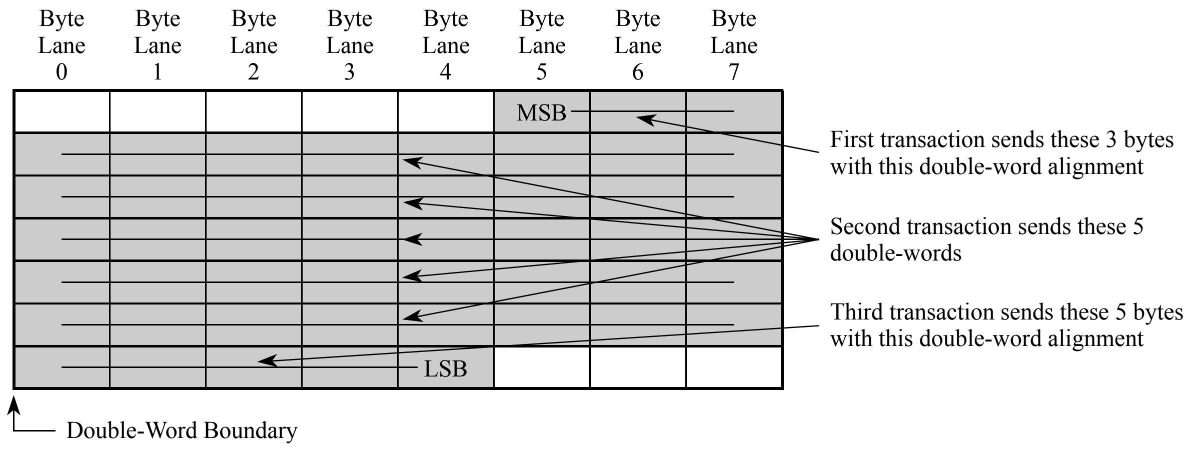

Byte Lane 0
Byte Lane 1
Byte Lane 2
Byte Lane 3
Byte Lane 4
Byte Lane 5
Byte Lane 6
Byte Lane 7
MSB
LSB
First transaction sends these 3 bytes with this double-word alignment
Second transaction sends these 5 double-words
Third transaction sends these 5 bytes with this double-word alignment
Double-Word Boundary

Figure 3-8. Data Alignment Example

Chapter 4 Packet Format Descriptions

4.1 Introduction

This chapter contains the packet format definitions for *Part 13: AMBA-RapidIO Scale-Out Logical Specification*. Reserved fields, unless defined in another logical specification, shall not be used by a device.

4.2 AMBA Fields

Figure 4-1 shows various fields related to coherent transactions in AMBA specification. These fields are summarized in Table 4-1.

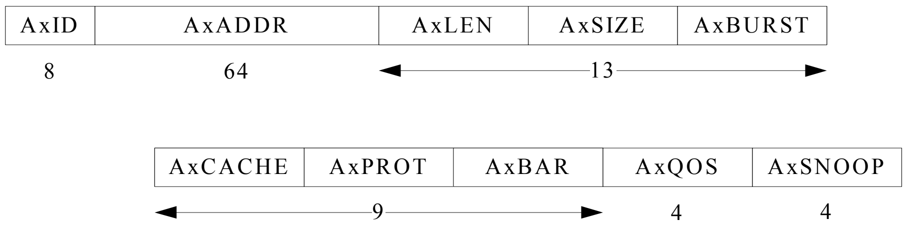


Figure 4-1. AMBA ACE Fields

Table 4-1. AMBA ACE Field Definition

Field	Definition
AxID	Address ID for read or write transaction. This field is the identification tag for the read or write address group of fields.
AxADDR	Address for read or write transaction. The field gives the address for the transfer.
AxLEN	Burst Length. This field identifies the exact number of transfers in a burst.
AxSIZE	Burst Size. This field indicates the size of each transfer in a burst.
AxBURST	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated.
AxCache	Memory type. This field indicates how transactions are required to progress through a system.
AxPROT	Protection type. This field indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
AxBAR	Barrier transaction. This field indicates a read or a write barrier transaction.
AxQOS	Quality of Service. The field identifies the QoS value for the read or the write transaction.
AxSNOOP	This field identifies the transaction type.

4.3 RapidIO Fields

Figure 4-2 shows the header fields for physical layer (PHY), transport layer and logical layer in a typical RapidIO packet.

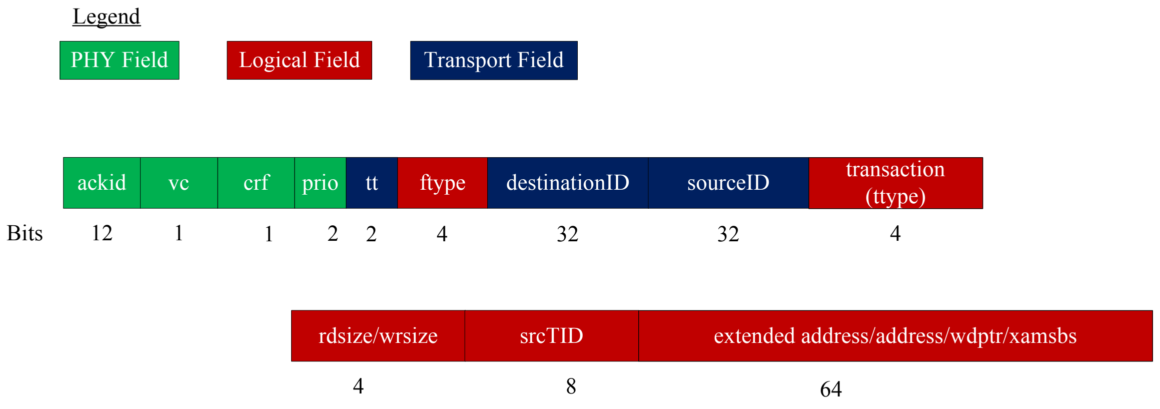


Figure 4-2. RapidIO Header Fields Packet Fields

To support transport the AMBA transactions over RapidIO the logical layer header fields are updated. Figure 4-3 shows all of the logical layer fields and Table 4-2 summarizes these fields.

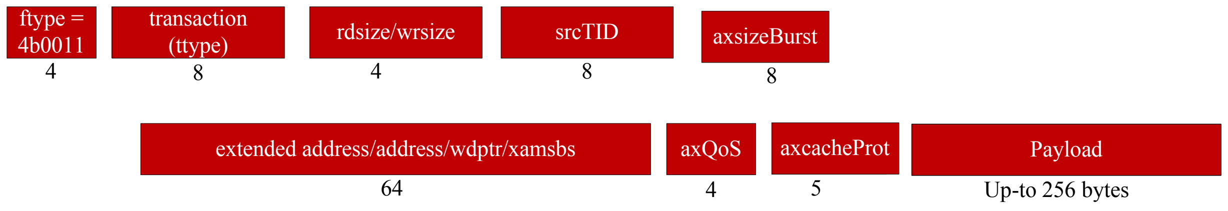


Figure 4-3. Scale-out Cache-Coherency Logical Layer Packet Fields

Table 4-2. Scale-out Cache-coherency Logical Layer Field Definition

Field	Definition
ftype	FType 3 (ftype = 0b0011) is used for the ARM-RapidIO Scale-out Cache Coherent Packets.
transaction (ttype)	8-bit ttype field is used to identify the transaction type. The field is divided into two sub-fields: TTYPE-H (3-bit) and TTYPE-T (5-bit)
rdsize	Data size for read transactions, used in conjunction with the word pointer (wdptr) bit.
wrsize	Write data size for sub-double-word transactions, used in conjunction with the word pointer (wdptr) bit. For writes greater than one double-word, the size is the maximum payload.
srcTID	The packet’s transaction ID.
extended address	Specifies the most significant 16 bits of a 50-bit physical address or 32 bits of a 66-bit physical address.
xamsbs	Extended address most significant bits. Further extends the address specified by the address and extended address fields by 2 bits. This field provides 34-, 50-, and 66-bit addresses to be specified in a packet with the xamsbs as the most significant bits in the address.
address	Least significant 29 bits (bits [0-28] of byte address [0-31]) of the double-word physical address.

Table 4-2. Scale-out Cache-coherency Logical Layer Field Definition (Continued)

Field	Definition
axsizeBurst	Identifies burst size.
axQoS	Quality of Service. The field identifies the QoS value for the RapidIO transactions.
axcacheProt	Identifies memory and protection type.

4.4 Field Mapping

The AMBA coherent transactions shall be mapped to the corresponding RapidIO fields to transport the AMBA transactions over RapidIO. Table 4-3 summarizes these field mappings.

Table 4-3. AMBA ACE and RapidIO Field Mapping

AMBA ACE		Part 13: AMBA-RapidIO Scale-Out Logical Specification Logical Layer Fields			
Field	Bits	Field	Bits	Total Bits	Notes
AxSNOOP	4	FTYPE	4	12	Use 4-bit FTYPE to identify AMBA-RAPIDIO Scale-out transactions and set FTYPE to 4’b0011. Use new 8-bit TTYPE field to identify the specific transaction types.
		TTYPE	8		
AxID	8	srcTID	8	8	Map AxID field to srcTID. AxID does not have to be unique. However, srcTID must be unique in a system.
AxADDR	64	Extended Address/Address/wdptr/xamsbs	64	64	Supports up-to 64-bit memory addressing using Extended Address/Address/wdptr/xamsbs.
AxLEN, AxSIZE, AxBURST	13	rdsize/wrsize	4	12	Identify read and write size using 4-bit rd/wrsize and 1-bit wdptr fields. Map AxLEN, AxSIZE, and AxBURST to 8-bit axsizeBurst field.
		axsizeBurst	8		
AxCACHE, AxPROT, AxBAR	9	axcacheProt	5	5	This field is used to communicate cache type, protection mechanism, and barrier transaction.
AxQOS	4	axQoS	4	8	4-bit axQoS field is used to transport AxQoS. 2-bit prio field is used to guarantee deadlock free communication. 2-bit <vc,crf> is used to support quality of service requirements through additional priority levels.
		vc, crf, prio	4		

4.5 Read and Write Size Definitions

Table 4-4. Read Size (rdsize) Definitions

wdptr	rdsize	Number of Bytes
0b0	0b0000	1
0b0	0b0001	1

wdptr	rdsize	Number of Bytes
0b0	0b0010	1
0b0	0b0011	1
0b1	0b0000	1
0b1	0b0001	1
0b1	0b0010	1
0b1	0b0011	1
0b0	0b0100	2
0b0	0b0101	3
0b0	0b0110	2
0b0	0b0111	5
0b1	0b0100	2
0b1	0b0101	3
0b1	0b0110	2
0b1	0b0111	5
0b0	0b1000	4
0b1	0b1000	4
0b0	0b1001	6
0b1	0b1001	6
0b0	0b1010	7
0b1	0b1010	7
0b0	0b1011	8
0b1	0b1011	16
0b0	0b1100	32
0b1	0b1100	64
0b0	0b1101	128
0b1	0b1101	256
0b0-1	0b1110 0b1111	Reserved

Table 4-5. Write Size (wrsize) Definitions

wdptr	rdsize	Number of Bytes
0b0	0b0000	1
0b0	0b0001	1
0b0	0b0010	1
0b0	0b0011	1
0b1	0b0000	1
0b1	0b0001	1
0b1	0b0010	1
0b1	0b0011	1

wdptr	rdsize	Number of Bytes
0b0	0b0100	2
0b0	0b0101	3
0b0	0b0110	2
0b0	0b0111	5
0b1	0b0100	2
0b1	0b0101	3
0b1	0b0110	2
0b1	0b0111	5
0b0	0b1000	4
0b1	0b1000	4
0b0	0b1001	6
0b1	0b1001	6
0b0	0b1010	7
0b1	0b1010	7
0b0	0b1011	8
0b1	0b1011	16
0b0	0b1100	32
0b1	0b1100	64
0b0	0b1101	128
0b1	0b1101	256
0b0-1	0b1110 0b1111	Reserved

4.6 Packet Format

Based on the fields discussed in earlier section, The ARM-RapidIO transactions shall use FType 3 (ftype = 0b0011) and the appropriate TTypes to identify the corresponding transactions as discussed in Section 4.7. A set of example packet format is shown in Figure 4-4 to Figure 4-7 for different payload and device sizes.

RapidIO-ARM Coherency Packet with Transport Type = Dev32 (64 Byte CG)

	+0	+1	+2	+3	
	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	
Byte 0	ackID, vc, CRF, prio	tt=10, Ftype	destination ID [31:16]		W 0
Byte 4	destination ID [15:0]		source ID [31:16]		W 1
Byte 8	source ID [15:0]		TType (transaction)	axQoS, rd-/wr-size	W 2
Byte 12	srcTID	axsizeBurst	axcacheProt, rsvd	rsvd	W 3
Byte 16	extended address [63:48]		extended address [47:32]		W 4
Byte 20	address [31:16]		address [15:3]	wdptr, xamsbs	W 5
Byte 24	Payload (Word 0)				W 6
Byte 28	Payload (Word 1)				W 7
Byte 32	Payload (Word 2)				W 8
Byte 36	Payload (Word 3)				W 9
Byte 40	Payload (Word 4)				W 10
Byte 44	Payload (Word 5)				W 11
Byte 48	Payload (Word 6)				W 12
Byte 52	Payload (Word 6)				W 13
Byte 56	Payload (Word 8)				W 14
Byte 60	Payload (Word 9)				W 15
Byte 64	Payload (Word 10)				W 16
Byte 68	Payload (Word 11)				W 17
Byte 72	Payload (Word 12)				W 18
Byte 76	Payload (Word 13)				W 19
Byte 80	CRC-16		Payload (Half-Word 14)		W 20
Byte 84	Payload (Half-Word 14)		Payload (Half-Word 15)		W 21
Byte 88	Payload (Half-Word 15)		CRC-16		W 22
Byte 92	CRC-32 (IDLE3 only)				W 23

Figure 4-4. RapidIO-ARM Coherency Packet Format with TType = Dev32 (64 Byte CG)

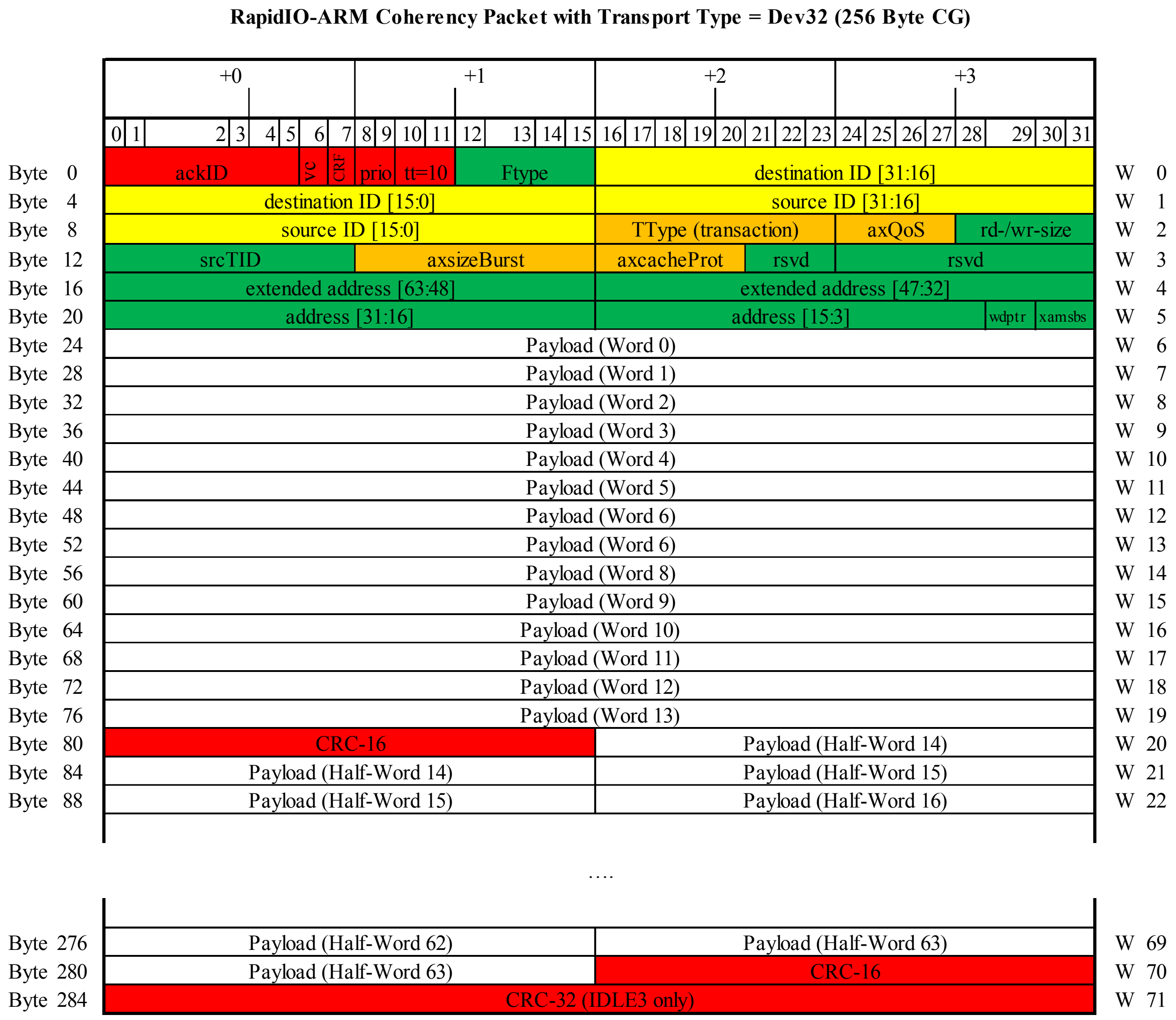

RapidIO-ARM Coherency Packet with Transport Type = Dev32 (256 Byte CG)
+0
+1
+2
+3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Byte 0: ackID | vc | CRF | prio | tt=10 | Ftype | destination ID [31:16] | W 0
Byte 4: destination ID [15:0] | source ID [31:16] | W 1
Byte 8: source ID [15:0] | TType (transaction) | axQoS | rd-/wr-size | W 2
Byte 12: srcTID | axsizeBurst | axcacheProt | rsvd | rsvd | W 3
Byte 16: extended address [63:48] | extended address [47:32] | W 4
Byte 20: address [31:16] | address [15:3] | wdptr | xamsbs | W 5
Byte 24: Payload (Word 0) | W 6
Byte 28: Payload (Word 1) | W 7
Byte 32: Payload (Word 2) | W 8
Byte 36: Payload (Word 3) | W 9
Byte 40: Payload (Word 4) | W 10
Byte 44: Payload (Word 5) | W 11
Byte 48: Payload (Word 6) | W 12
Byte 52: Payload (Word 6) | W 13
Byte 56: Payload (Word 8) | W 14
Byte 60: Payload (Word 9) | W 15
Byte 64: Payload (Word 10) | W 16
Byte 68: Payload (Word 11) | W 17
Byte 72: Payload (Word 12) | W 18
Byte 76: Payload (Word 13) | W 19
Byte 80: CRC-16 | Payload (Half-Word 14) | W 20
Byte 84: Payload (Half-Word 14) | Payload (Half-Word 15) | W 21
Byte 88: Payload (Half-Word 15) | Payload (Half-Word 16) | W 22
….
Byte 276: Payload (Half-Word 62) | Payload (Half-Word 63) | W 69
Byte 280: Payload (Half-Word 63) | CRC-16 | W 70
Byte 284: CRC-32 (IDLE3 only) | W 71

Figure 4-5. RapidIO-ARM Coherency Packet Format with TType = Dev32 (256 Byte CG)

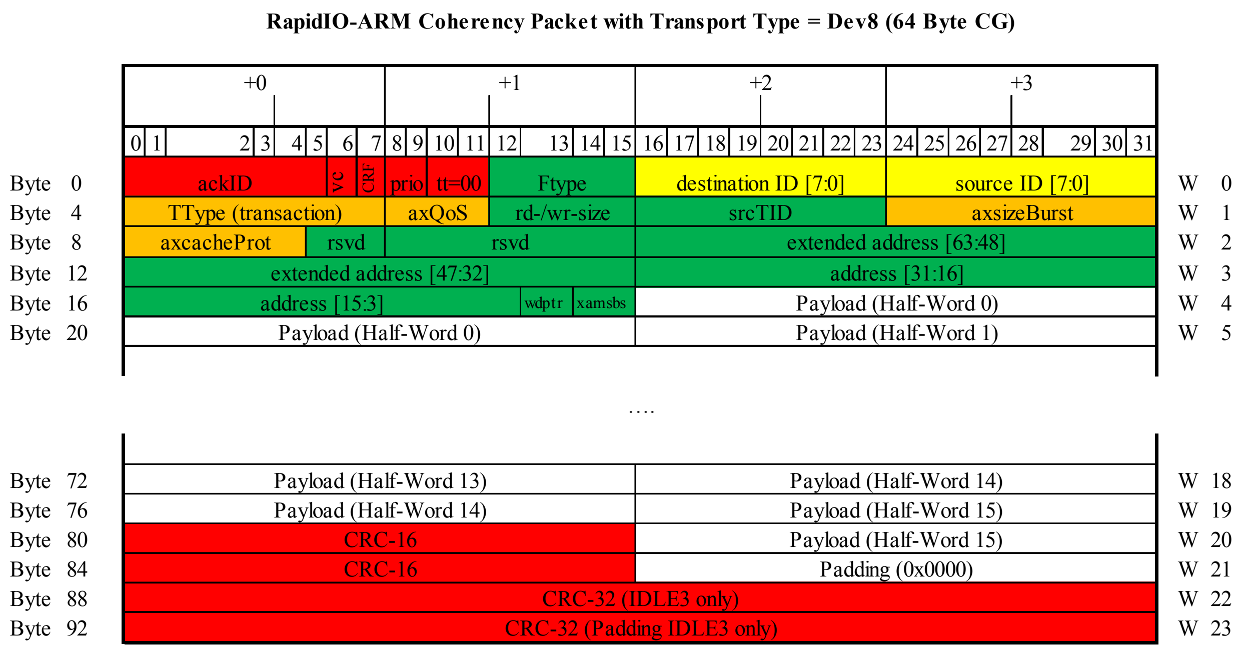

RapidIO-ARM Coherency Packet with Transport Type = Dev8 (64 Byte CG)
+0
+1
+2
+3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Byte 0 | ackID | VC | CRF | prio | tt=00 | Ftype | destination ID [7:0] | source ID [7:0] | W 0
Byte 4 | TType (transaction) | axQoS | rd-/wr-size | srcTID | axsizeBurst | W 1
Byte 8 | axcacheProt | rsvd | rsvd | extended address [63:48] | W 2
Byte 12 | extended address [47:32] | address [31:16] | W 3
Byte 16 | address [15:3] | wdptr | xamsbs | Payload (Half-Word 0) | W 4
Byte 20 | Payload (Half-Word 0) | Payload (Half-Word 1) | W 5
….
Byte 72 | Payload (Half-Word 13) | Payload (Half-Word 14) | W 18
Byte 76 | Payload (Half-Word 14) | Payload (Half-Word 15) | W 19
Byte 80 | CRC-16 | Payload (Half-Word 15) | W 20
Byte 84 | CRC-16 | Padding (0x0000) | W 21
Byte 88 | CRC-32 (IDLE3 only) | W 22
Byte 92 | CRC-32 (Padding IDLE3 only) | W 23

Figure 4-6. RapidIO-ARM Coherency Packet Format with TType = Dev8 (64 Byte CG)

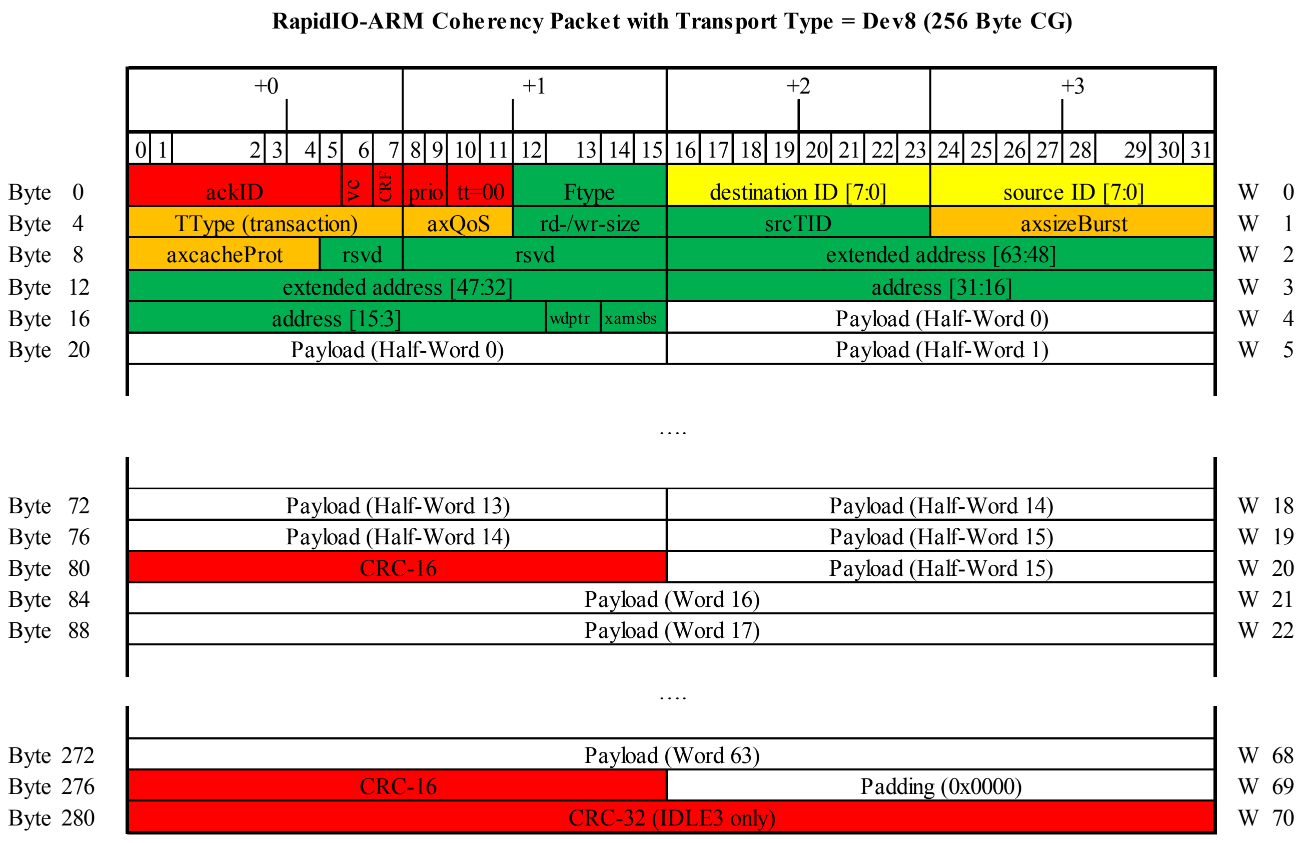

RapidIO-ARM Coherency Packet with Transport Type = Dev8 (256 Byte CG)
+0
+1
+2
+3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Byte 0 | ackID | VC | CRF | prio | tt=00 | Ftype | destination ID [7:0] | source ID [7:0] | W 0
Byte 4 | TType (transaction) | axQoS | rd-/wr-size | srcTID | axsizeBurst | W 1
Byte 8 | axcacheProt | rsvd | rsvd | extended address [63:48] | W 2
Byte 12 | extended address [47:32] | address [31:16] | W 3
Byte 16 | address [15:3] | wdptr | xamsbs | Payload (Half-Word 0) | W 4
Byte 20 | Payload (Half-Word 0) | Payload (Half-Word 1) | W 5
….
Byte 72 | Payload (Half-Word 13) | Payload (Half-Word 14) | W 18
Byte 76 | Payload (Half-Word 14) | Payload (Half-Word 15) | W 19
Byte 80 | CRC-16 | Payload (Half-Word 15) | W 20
Byte 84 | Payload (Word 16) | W 21
Byte 88 | Payload (Word 17) | W 22
….
Byte 272 | Payload (Word 63) | W 68
Byte 276 | CRC-16 | Padding (0x0000) | W 69
Byte 280 | CRC-32 (IDLE3 only) | W 70

Figure 4-7. RapidIO-ARM Coherency Packet Format with TType = Dev8 (256 Byte CG)

4.7 Field Encoding

The encoding between the AMBA ACE transactions and the 8-bit TType field is shown in Table 4-6. The TType filed is divided into two parts: TType-Header (TType-H) and TType-Tail (TType-T). The most significant 3-bit TType-H field identifies the transaction group and the least significant 5-bit TType-T field identifies the specific transaction in that group.

Field	Tranaction (TType) [7:0]		Transaction Name	RM Transaction Grou
	TType-H	TType-T		
	[7:5]	[4:0]		
Read	0x0	0x0	ReadNoSnoop	Non-Snooping
		0x1	ReadOnce	Coherent
		0x2	ReadShared	
		0x3	ReadClean	
		0x4	ReadNotSharedDirty	
		0x5	ReadUnique	
		0x6-0x1F	Reserved	
Write	0x1	0x0	WriteNoSnoop	Non-Snooping
		0x1	WriteUnique	Coherent
		0x2	WriteLineUnique	
		0x3	WriteBack	MemoryUpdate
		0x4	WriteClean	
		0x5-0x1F	Reserved	
Snoop	0x2	0x0	SnoopReadOnce	Snoop
		0x1	SnoopReadClean	
		0x2	SnoopReadShared	
		0x3	SnoopReadNotSharedDirty	
		0x4	SnoopReadUnique	
		0x5	SnoopCleanInvalid	
		0x6	SnoopMakeInvalid	
		0x7	SnoopCleanShared	
		0x8-0x1F	Reserved	
No Payload	0x3	0x0	CleanUnique	Coherent
		0x1	MakeUnique	
		0x2	CleanShared	CacheMaintenance and Memory Update
		0x3	CleanInvalid	
		0x4	MakeInvalid	
		0x5	Evict	
		0x6-0x1F	Reserved	
DVM	0x4	0x0	DVMComplete	DVM
		0x1	DVMMessage	
		0x2-0x1F	Reserved	
Atomics	0x5	0x0-0x1F	TBC	Atomics
ResponseData	0x6	0x0	ReadResponse	
		0x1	SnoopResponse	
		0x2-0x1F	Reserved	
ResponseNoData	0x7	0x0	WriteResponse	
		0x1	SnoopResponse	
		0x2	DataLessResponse	
		0x3-0x1F	Reserved	

Table 4-6. ARM Coherency Transaction to TType Field Encoding

The encoding format for the Atomic transactions is reserved for the AMBA5 CHI specification.

Chapter 5 Scale-out Coherency Registers

5.1 Introduction

This chapter describes the visible register set that allows an external processing element to determine the capabilities, configuration, and status of a processing element using this logical specification. This chapter describes registers or register bits only defined by this specification. To determine a complete list of registers and bit definitions, refer to the other RapidIO logical, transport, and physical specifications. All registers are 32 bits and aligned to a 32-bit boundary.

5.2 Register Summary

Table 5-1 shows the register map for this RapidIO specification. These capability registers (CARs) and command and status registers (CSRs) can be accessed using *RapidIO Part 1: Input/Output Logical Specification* maintenance operations. Any register offsets not defined are considered reserved for this specification unless otherwise stated. Other registers required for a processing element are defined in other applicable RapidIO specifications and by the requirements of the specific device, and are beyond the scope of this specification. Read and write accesses to reserved register offsets shall terminate normally and not cause an error condition in the target device. Writes to CAR (read-only) space shall terminate normally and not cause an error condition in the target device.

Register bits defined as reserved are considered reserved for this specification only. Bits that are reserved in this specification may be defined in another RapidIO specification.

Table 5-1. Scale-out Register Map

Configuration Space Byte Offset	Register Name
0x0-14	Reserved
0x18	Source Operations CAR
0x1C	Destination Operations CAR
0x20-FC	Reserved

Table 5-1. Scale-out Register Map (Continued)

Configuration Space Byte Offset	Register Name
0x100-FFFC	Extended Features Space
0x10000-FFFFFC	Implementation-defined Space

5.3 Reserved Register, Bit and Bit Field Value Behavior

Table 5-2 describes the required behavior for accesses to reserved register bits and reserved registers for the RapidIO register space.

Table 5-2. Configuration Space Reserved Access Behavior

Byte Offset	Space Name	Item	Initiator behavior	Target behavior
0x0-3C	Capability Register Space (CAR Space - this space is read-only)	Reserved bit	read - ignore returned value[1]	read - return logic 0
			write	write - ignored
		Implementation-defined bit	read - ignore returned value unless implementation-defined function understood	read - return implementation-defined value
			write	write - ignored
		Reserved register	read - ignore returned value	read - return logic 0s
			write	write - ignored
0x40-FC	Command and Status Register Space (CSR Space)	Reserved bit	read - ignore returned value	read - return logic 0
			write - preserve current value[2]	write - ignored
		Implementation-defined bit	read - ignore returned value unless implementation-defined function understood	read - return implementation-defined value
			write - preserve current value if implementation-defined function not understood	write - implementation-defined
		Reserved register	read - ignore returned value	read - return logic 0s
			write	write - ignored

Table 5-2. Configuration Space Reserved Access Behavior (Continued)

Byte Offset	Space Name	Item	Initiator behavior	Target behavior
0x100-FFFC	Extended Features Space	Reserved bit	read - ignore returned value	read - return logic 0
			write - preserve current value	write - ignored
		Implementation-defined bit	read - ignore returned value unless implementation-defined function understood	read - return implementation-defined value
			write - preserve current value if implementation-defined function not understood	write - implementation-defined
		Reserved register	read - ignore returned value	read - return logic 0s
			write	write - ignored
0x10000-FFFFFC	Implementation-defined Space	Reserved bit and register	All behavior implementation-defined	

[1]Do not depend on reserved bits being a particular value; use appropriate masks to extract defined bits from the read value.

[2]All register writes shall be in the form: read the register to obtain the values of all reserved bits, merge in the desired values for defined bits to be modified, and write the register, thus preserving the value of all reserved bits.

When a writable bit field is set to a reserved value, device behavior is implementation specific.

5.4 Capability Registers (CARs)

Every processing element shall contain a set of registers that allows an external processing element to determine its capabilities using the I/O logical maintenance read operation. All registers are 32 bits wide and are organized and accessed in 32-bit (4 byte) quantities, although some processing elements may optionally allow larger accesses. CARs are read-only. For the required behavior for accesses to reserved registers and register bits, see Table 5-2.

CARs are big-endian with bit 0 the most significant bit.

5.4.1 Source Operations CAR (Configuration Space Offset 0x18)

This register defines the set of RapidIO Scale-out logical operations that can be issued by this processing element (see Table 5-3). It is assumed that a processing element can generate I/O logical maintenance read and write requests if it is required to access CARs and CSRs in other processing elements. RapidIO switches shall be able to route any packet.

Table 5-3. Bit Settings for Source Operations CAR

Bit	Field Name	Description
0-9	—	Reserved
10	ACE Coherency Support	PE can originate AMBA Coherency transactions
11-13	—	Reserved
14–15	Implementation defined	Defined by the device implementation
16–29	—	Reserved
30–31	Implementation defined	Defined by the device implementation

5.4.2 Destination Operations CAR (Configuration Space Offset 0x1C)

This register defines the set of RapidIO Scale-out operations that can be supported by this processing element (see Table 5-4). It is required that all processing elements can respond to I/O logical maintenance read and write requests in order to access these registers. This register is applicable for endpoint devices only. RapidIO switches shall be able to route any packet.

Table 5-4. Bit Settings for Destination Operations CAR

Bit	Field Name	Description
0-9	—	Reserved
10	ACE Coherency Support	PE can process ARM Coherency transactions
11-13	—	Reserved
14-15	Implementation defined	Defined by the device implementation
16-29	—	Reserved
30-31	Implementation defined	Defined by the device implementation

5.5 Command and Status Registers (CSRs)

The *64-bit ARM-RapidIO Scale-out Logical Specification* does not define any command and status registers (CSRs).

5.6 AMBA RapidIO Scale-Out Register Block (CSRs)

Are there any functions that should be placed in this block?

Extension of source/destination capabilities CSRs, specific to ARM64 cache coherency transactions?

Standardized controls/programming model?

Anything for mutex/initialization of local/remote directories?

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book.

A

Address collision. An address based conflict between two or more cache coherence operations when referencing the same coherence granule.

B

Big-endian. A byte-ordering method in memory where the address n of a word corresponds to the most significant byte. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the most significant byte.

Broadcast. The concept of sending a packet to all processing elements in a system.

C

Cache. High-speed memory containing recently accessed data and/or instructions (subset of main memory) associated with a processor.

Cache coherence. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor's cache. In other words, a write operation to an address in the system is visible to all other caches in the system. Also referred to as memory coherence.

Cache coherent-non uniform memory access (**CC-NUMA**). A cache coherent system in which memory accesses have different latencies depending upon the physical location of the accessed address.

Capability registers (CARs). A set of read-only registers that allows a processing element to determine another processing element's capabilities.

Coherence domain. A logically associated group of processing elements that participate in the globally shared memory protocol and are able to maintain cache coherence among themselves.

Coherence granule. A contiguous block of data associated with an address for the purpose of guaranteeing cache coherence.

Command and status registers (CSRs). A set of registers that allows a processing element to control and determine the status of another processing element’s internal hardware.

D

Deadlock. A situation in which two processing elements that are sharing resources prevent each other from accessing the resources, resulting in a halt of system operation.

Destination. The termination point of a packet on the RapidIO interconnect, also referred to as a target.

Device. A generic participant on the RapidIO interconnect that sends or receives RapidIO transactions, also called a processing element.

Device ID. The identifier of an end point processing element connected to the RapidIO interconnect.

Distributed memory. System memory that is distributed throughout the system, as opposed to being centrally located.

Domain. A logically associated group of processing elements.

Double-word. An eight byte quantity, aligned on eight byte boundaries.

E

End point. A processing element which is the source or destination of transactions through a RapidIO fabric.

F

Field or Field name. A sub-unit of a register, where bits in the register are named and defined.

H

Half-word. A two byte or 16 bit quantity, aligned on two byte boundaries.

Home memory. The physical memory corresponding to the physical address of a coherence granule.

I

Initiator. The origin of a packet on the RapidIO interconnect, also referred to as a source.

Invalidate operation. An operation used to remove a coherence granule from caches within the coherence domain.

L

Little-endian. A byte-ordering method in memory where the address n of a word corresponds to the least significant byte. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the most significant byte.

Local memory. Memory associated with the processing element in question.

LSB. Least significant byte.

M

Memory coherence. Memory is coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor's cache. In other words, a write operation to an address in the system is visible to all other caches in the system. Also referred to as cache coherence.

Memory controller. The point through which home memory is accessed.

Memory directory. A table of information associated with home memory that is used to track the location and state of coherence granules cached by coherence domain participants.

Message passing. An application programming model that allows processing elements to communicate via messages to mailboxes instead of via DMA or GSM. Message senders do not write to a memory address in the receiver.

MSB. Most significant byte.

Multicast. The concept of sending a packet to more than one processing elements in a system.

N

Non-coherent. A transaction that does not participate in any system globally shared memory cache coherence mechanism.

O

Operation. A set of transactions between end point devices in a RapidIO system (requests and associated responses) such as a read or a write.

P

Packet. A set of information transmitted between devices in a RapidIO system.

Priority. The relative importance of a packet; in most systems a higher priority packet will be serviced or transmitted before one of lower priority.

Processing Element (**PE**). A generic participant on the RapidIO interconnect that sends or receives RapidIO transactions, also called a device.

Processor. The logic circuitry that responds to and processes the basic instructions that drive a computer.

R

Read operation. An operation used to obtain a globally shared copy of a coherence granule.

Read-for-ownership operation. An operation used to obtain ownership of a coherence granule for the purposes of performing a write operation.

Remote access. An access by a processing element to memory located in another processing element.

Sharing mask. The state associated with a coherence granule in the memory directory that tracks the processing elements that are sharing the coherence granule.

Source. The origin of a packet on the RapidIO interconnect, also referred to as an initiator.

Sub-double-word. Aligned on eight byte boundaries.

Switch. A multiple port processing element that directs a packet received on one of its input ports to one of its output ports.

T

Target. The termination point of a packet on the RapidIO interconnect, also referred to as a destination.

Transaction. A specific request or response packet transmitted between end point devices in a RapidIO system.

W

Write-through. A cache policy that passes all write operations through the caching hierarchy directly to home memory.

Word. A four byte or 32 bit quantity, aligned on four byte boundaries.