Ncore 3.2 System Architecture Specification

Rev: 0.52, April 3, 2022

Arteris® Ncore 3.2 System Architecture Specification

Release Information

Version	*Editor*	*Change*	*Date*
0.5	MK/MF	Initial Document created from older original spec docs	03/25/2022
0.52	MK	Updated power and clock section Added SRAM protection to error section	04/02/2022
Legend:	MK MF Xx	Mohammed Michael Frank Whoever else edited this document	

Confidential Proprietary Notice

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information: (i) for the purposes of determining whether implementations infringe any third party patents;(ii) for developing technology or products which avoid any of Arteris IP's intellectual property; or (iii) as a reference for modifying existing patents or patent applications or creating any continuation, continuation in part, or extension of existing patents or patent applications; or (iv) for generating data for publication or disclosure to third parties, which compares the performance or functionality of the Arteris IP technology described in this document with any other products created by you or a third party, without obtaining Arteris IP's prior written consent.

THIS DOCUMENT IS PROVIDED “AS IS”. ARTERIS IP PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arteris IP makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights. This document may include technical inaccuracies or typographical errors. Arteris IP makes no representations or warranties against the risk or presence of same.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARTERIS IP BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARTERIS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be solely responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arteris IP's customers is not intended to create or refer to any partnership relationship with any other company. Arteris IP may make changes to this document at any time and without notice. If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arteris IP, then the click-through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the agreement shall prevail.

Confidentiality Status

This document is Confidential and Proprietary. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arteris IP and the party that Arteris IP delivered this document to.

Product Status

The information in this document is ***Preliminary***.

Web Address

http://www.arteris.com

Table of Contents

Table of Figures

Table of Tables

Preface

This preface introduces the Arteris® Network-on-Chip Hierarchical Coherency Engine Architecture Specification.

About this document

This technical document is for the Arteris Network-on-Chip Hierarchical Coherency Engine Architecture. It describes the subsystems and their function along with the system's interactions with the external subsystems. It also provides reference documentation and contains programming details for registers.

Product revision status

TBD

Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses or intend to use the Arteris Network-on-Chip Hierarchical Coherency System (ANoC-HCS).

Using this document

TBD

Glossary

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

Typographic conventions

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 italic

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.

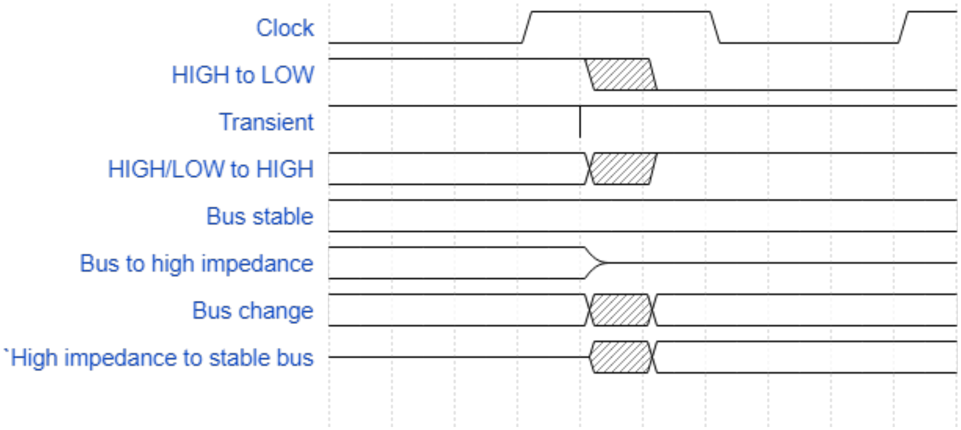
SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the Arteris® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

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



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.

Lowercase n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

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

History of the World II, Mel Brooks.

1 Introduction

The Architectural system Overview with different concerto units is shown in Figure 1. A concerto unit can expose upto three different main interface types, they are classified as

- Native interface: this is for communication with other standard interface IPs
- SMI: this is for concerto protocol communication
- APB: this is for control and status register (CSR) communication

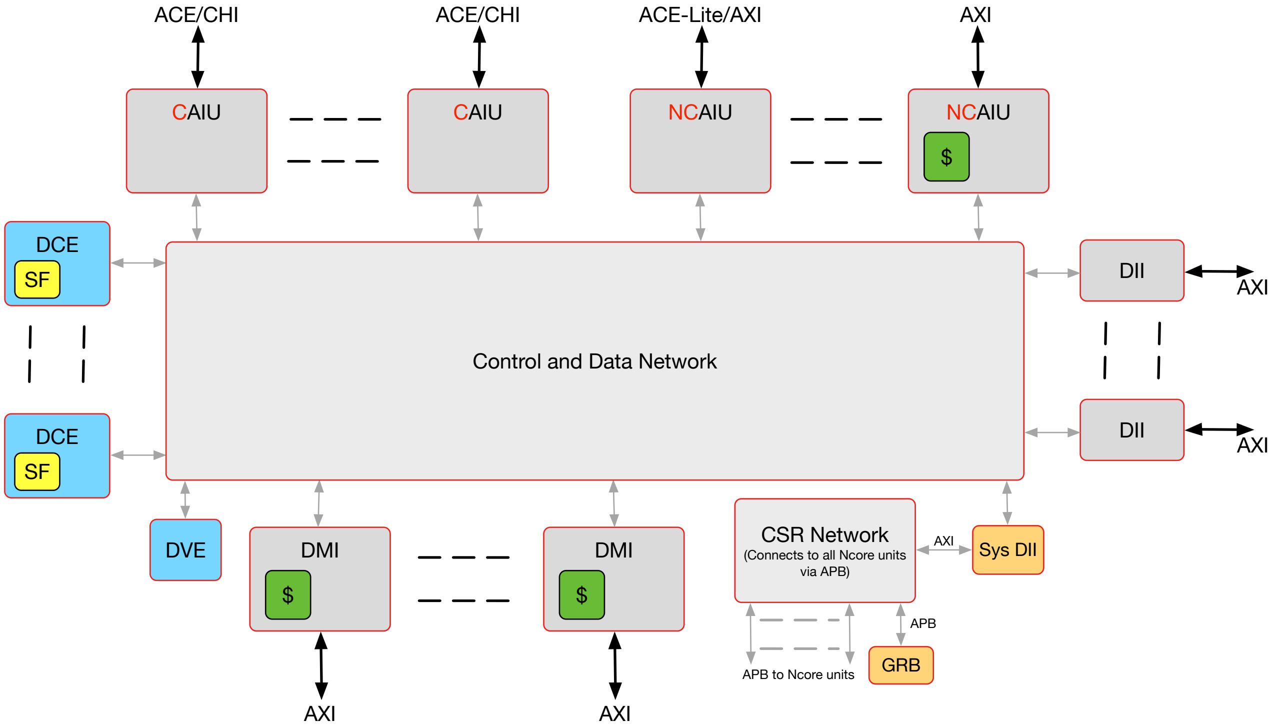


FIGURE 1 SYSTEM OVERVIEW

For the sake of brevity, the Figure 1 is simplified and does not show complete details of the CSR network and APB connections to all concerto units. The CSR network is not within the scope of this document, please refer to system architecture document for details. The different concerto units shown are described below

1.1 Concerto Units

Agent Interface Unit (AIU) connects to an Agent that performs coherent accesses. Each AIU exposes a single native interface. This architecture specifies a different AIU for each type of native interface, based on the type of the native interface an AIU is further classified as either coherent agent interface unit (CAIU) or non-coherent agent interface unit (NCAIU). CAIU connects to an agent that has a coherent snoopable cache while NCAIU connects to an agent that does not have a coherent snoopable cache. An NCAIU may also be referred to as IOAIU; for the sake of brevity in this document when both CAIU and NCAIU are to be referred together, they are referred as simply AIU.

An NCAIU can be configured to implement a coherent snoopable cache within itself, this cache is referred to as proxy cache.

Distributed Coherency Engine (DCE) is responsible for keeping copies of cacheline coherent. Each cacheline address is associated with a single DCE, known as the Home DCE for that cacheline address. The Home DCE acts as the point of serialization for, and enforces coherence on, the given cacheline address. It is also responsible for preserving the order of accesses from Concerto units to a given cacheline address and determining the visibility of data from those accesses; in particular, a DCE represents the coherence domain visibility point and controls access to the system visibility point. In a system configured with multiple DCEs, cacheline address are distributed across the DCEs using an implementation-defined mapping function, e.g. address bit interleaving or hashing.

Distributed Virtual Engine (DVE) is responsible for broadcasting DVM messages.

Distributed Memory Interface (DMI) connects to system memory or memory controller with a master native interface like AXI. Each cacheline address is associated with a single DMI, known as the Home DMI for that cacheline address. A system can be configured with multiple DMIs, cacheline address are distributed across the DMIs using an implementation-defined mapping function, e.g. address bit interleaving or hashing. The distribution of cacheline addresses across DMIs may differ from the distribution across DCEs; for example, DMI address bit interleaving may be different from DCE address bit interleaving. A DMI may be configured with a System Memory Cache (SMC).

Distributed IO Interface (DII) connects to peripheral or IO devices via a master native interface like AXI. This provides a path for connected agents to access Non-system memory or peripheral address space.

Global Register Block (GRB) Connects only to the CSR network and hosts system identification registers

The transport interconnect (Legato) also known as control and data network (CDN) may consist of one or more networks. CDN connects to other concerto units via a Symphony Messaging Interface (SMI), which encapsulates concerto messages as shown in Table 1.

SMI NDP Mapping	
SMI	**Concerto Message**
ndp_targ_id	{target_id, fPortId}
ndp_src_id	{initiator_id, fPortId}
ndp_steer	steering
ndp_msg_tier	t_tier
ndp_msg_qos	ql
ndp_pri	priority
ndp_len	wNdpBits parameter
ndp_ndp	See cpr/concMsg. This can include m_prot if SMI message protection is enabled. Note wMProt is based on the NDP width per message type.
ndp_dp_present	1 for DTWReq and DTRReq, otherwise 0
ndp_msg_id	message_id
ndp_msg_user	{Header protection bits (if enable SMI header protection) }
ndp_msg_err	0 (unused)
SMI DP Mapping	
SMI	**Concerto Message**

dp_last	last
dp_data	data
dp_user	NUM_DW_PER_BEAT x {dbad, dwid, dprot(if enable SMI data protection), be, user}

Table 1 SMI to Concerto Message Mapping

2 Transport Interconnect

Ncore uses two types of transport interconnect: the Control and Status Transport Interconnect (CSTI) and Control and Data Transport Interconnect (CDTI).

CSTI is the service interconnect for configuration and status register access to all Ncore components. CDTI is the main protocol interconnect.

Ncore components generate different set of control and data messages as described in Concerto protocol specification. These messages are mapped to one data network and at least two control network, these networks are referred together as CDTI. Separation of different messages onto independent transport resources avoids dependency loops inside the transport that would otherwise have deadlock issues and may also improve performance.

The number of control networks are determined based on floor plan and throughput requirements. Control network width is determined based on system address width and other protocol control information that needs to flow on it. Data network supports three width options, 64, 128, and 256 bit. These options can be mixed and matched within a single data network.

For a single native request, Ncore protocol issues multiple control messages and most likely a single data message. Depending on the width of the data network, one or more control network can be chosen to achieve the required throughput. For example, If the data network is 256bits, a cache line read will consume 2 data network cycles. In this case to achieve maximum throughput three control networks must be used. This ensures the number of control network cycles required will be within the 2 data network cycles. Currently, Ncore supports the following two options:

- Three networks, one data and two control networks, one for requests and another for responses
- Four networks, one data and three control networks, one for requests,another for responses and lastly a dedicated network for messages between DCEs and DMIs.

2.1 CDTI Network

This section goes over CDTI connectivity. In the description following, CN0 is for control network 0, CN1 is for control network 1, CN2 is for control network 2 and DN is for data network.

The mapping for the control and data network in Four network, 3CN1DN configuration is shown in Table 2. Three networks, 2CN1DN configuration is achieved by combining CN1 and CN2

Unit	Control Network 0 (CN0)		Control Network 1 (CN1)		Control Network 2 (CN2)		Data Network (DN)	
	Tx Msg	Rx Msg	Tx Msg	Rx Msg	Tx Msg	Rx Msg	Tx Msg	Rx Msg
CAIU								
	CmdReq	StrReq	StrRsp	CmdRsp			DtwReq/ DtwDbgReq	DtrReq
	SysReq	SnpReq	SnpRsp	DtrRsp			DtrReq	
		SysReq	DtrRsp	CmpRsp				
			SysRsp	DtwRsp/ DtwDbgRsp				
				SysRsp				
NCAIU (ACE-Lite/ACE-Lite E with DVM)								
	CmdReq	StrReq	StrRsp	CmdRsp			DtwReq/ DtwDbgReq	DtrReq
	SysReq	SnpReq	SnpRsp	DtrRsp			DtrReq	
			DtrRsp	DtwRsp/ DtwDbgRsp				
				SysRsp				
NCAIU (AXI with proxy cache)								
	CmdReq	StrReq	StrRsp	CmdRsp			DtwReq/ DtwDbgReq	DtrReq
	SysReq	SnpReq	SnpRsp	DtrRsp			DtrReq	
		SysReq	DtrRsp	DtwRsp/ DtwDbgRsp				
			SysRsp	SysRsp				
NCAIU (AXI without proxy cache and ACE-Lite/ACE-Lite E without DVM)								
	CmdReq	StrReq	StrRsp	CmdRsp			DtwReq/ DtwDbgReq	DtrReq
		SnpReq	SnpRsp	DtrRsp			DtrReq	
			DtrRsp	DtwRsp/ DtwDbgRsp				
DCE								
	StrReq	CmdReq	CmdRsp	SnpRsp	MrdReq	MrdRsp		
	SnpReq	SysReq	SysRsp	StrRsp	RbrReq	RbrRsp		
	SysReq			SysRsp	RbuRsp	RbuReq		
DII								
	StrReq	CmdReq	CmdRsp	StrRsp			DtrReq	DtwReq
			DtwRsp	DtwDbgRsp			DtwDbgReq	
				DtrRsp				
DMI								
	StrReq	CmdReq	CmdRsp	StrRsp	MrdRsp	MrdReq	DtrReq	DtwReq
			DtwRsp	DtwDbgRsp	RbrRsp	RbrReq	DtwDbgReq	
				DtrRsp	RbuReq	RbuRsp		
DVE								
	StrReq	CmdReq	CmdRsp	StrRsp				DtwReq/ DtwDbgReq
	SnpReq	SysReq	CmpRsp	SnpRsp				
			DtwRsp/ DtwDbgRsp					
			SysRsp					

Table 2 System Control and Data Network Mapping

Connectivity between different Ncore units is shown in Table 3. The top row and the left most column specify the Ncore unit names, the intersection point specifies the networks that connect the two units. Intersection points that are empty signifies that there are no connections between the corresponding units.

	CAIU	NCAIU	DCE	DII	DMI	DVE
CAIU	DN CN1	DN CN1	CN0 CN1	CN0 CN1 DN	CN0 CN1 DN	CN0 CN1 DN
NCAIU	DN CN1	DN CN1	CN0 CN1	CN0 CN1 DN	CN0 CN1 DN	CN0 CN1 DN
DCE	CN0 CN1	CN0 CN1			CN2	
DII	CN0 CN1 DN	CN0 CN1 DN				
DMI	CN0 CN1 DN	CN0 CN1 DN	CN2			
DVE	CN0 CN1 DN	CN0 CN1 DN				

TABLE 3 CONNECTIVITY MAPPING

Detailed TX and RX connectivity between Ncore units for different networks are shown in Table 4, Table 5, Table 6 and Table 7. The top row and the left most column specify the Ncore unit names, the intersection point specifies the connectivity. The direction of connectivity is from column name perspective, i.e. the port names TX/RX specifies the connectivity on the column unit.

	CAIU CHI	CAIU ACE	NCAIU AXI W Cache	NCAIU AXI No Cache NC =0	NCAIU AXI No Cache NC =1	NCAIU ACE-Lite	NCAIU ACE-Lite-E	DCE	DII	DMI	DVE
CAIU CHI								TX/RX	TX/RX	TX/RX	TX/RX
CAIU ACE								TX/RX	TX/RX	TX/RX	TX/RX
NCAIU AXI W Cache								TX/RX		TX/RX	
NCAIU AXI No Cache NC =0								TX/RX		TX/RX	
NCAIU AXI No Cache NC =1									TX/RX	TX/RX	
NCAIU ACE-Lite								TX/RX	TX/RX	TX/RX	TX/RX(if DVM both)
NCAIU ACE-Lite-E								TX/RX	TX/RX	TX/RX	TX/RX(if DVM both)
DCE	TX/RX	TX/RX	TX/RX	TX/RX		TX/RX	TX/RX				
DII	TX/RX	TX/RX			TX/RX	TX/RX	TX/RX				

	CAIU CHI	CAIU ACE	NCAIU AXI W Cache	NCAIU AXI No Cache NC =0	NCAIU AXI No Cache NC =1	NCAIU ACE-Lite	NCAIU ACE-Lite-E	DCE	DII	DMI	DVE
DMI	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX				
DVE	TX/RX	TX/RX				TX/RX (if DVM both)	TX/RX (if DVM both)				

TABLE 4 CN0 TX RX CONNECTIVITY MAP

	CAIU CHI	CAIU ACE	NCAIU AXI W Cache	NCAIU AXI No Cache NC =0	NCAIU AXI No Cache NC =1	NCAIU ACE-Lite	NCAIU ACE-Lite-E	DCE	DII	DMI	DVE
CAIU CHI	TX/RX	TX/RX	TX/RX	TX		TX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX
CAIU ACE	TX/RX	TX/RX	TX/RX	TX		TX	TX	TX/RX	TX/RX	TX/RX	TX/RX
NCAIU AXI W Cache	TX/RX	TX/RX	TX/RX	TX		TX	TX	TX/RX		TX/RX	TX
NCAIU AXI No Cache NC =0	RX	RX	RX					TX/RX		TX/RX	TX
NCAIU AXI No Cache NC =1									TX/RX	TX/RX	TX
NCAIU ACE-Lite	RX	RX	RX					TX/RX	TX/RX	TX/RX	TX/RX(if DVM)
NCAIU ACE-Lite-E	TX/RX	RX	RX					TX/RX	TX/RX	TX/RX	TX/RX(if DVM)
DCE	TX/RX	TX/RX	TX/RX	TX/RX		TX/RX	TX/RX				
DII	TX/RX	TX/RX			TX/RX	TX/RX	TX/RX				TX
DMI	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX				TX
DVE	TX/RX	TX/RX	RX	RX	RX	TX(if DVM)/RX	TX(if DVM)/RX		RX	RX	

TABLE 5 CN1 TX RX CONNECTIVITY MAP

	CAIU CHI	CAIU ACE	NCAIU AXI W Cache	NCAIU AXI No Cache NC =0	NCAIU AXI No Cache NC =1	NCAIU ACE-Lite	NCAIU ACE-Lite-E	DCE	DII	DMI	DVE
CAIU CHI											
CAIU ACE											
NCAIU AXI W Cache											
NCAIU AXI No Cache NC =0											
NCAIU AXI No Cache NC =1											
NCAIU ACE-Lite											
NCAIU ACE-Lite-E											
DCE										TX/RX	
DII											
DMI								TX/RX			
DVE											

TABLE 6 CN2 TX RX CONNECTIVITY MAP

	CAIU CHI	CAIU ACE	NCAIU AXI W Cache	NCAIU AXI No Cache NC =0	NCAIU AXI No Cache NC =1	NCAIU ACE-Lite	NCAIU ACE-Lite-E	DCE	DII	DMI	DVE
CAIU CHI	TX/RX	TX/RX	TX/RX	RX		RX	TX/RX		TX/RX	TX/RX	RX
CAIU ACE	TX/RX	TX/RX	TX/RX	RX		RX	RX		TX/RX	TX/RX	RX
NCAIU AXI W Cache	TX/RX	TX/RX	TX/RX	RX		RX	RX			TX/RX	RX
NCAIU AXI No Cache NC =0	TX	TX	TX							TX/RX	RX
NCAIU AXI No Cache NC =1									TX/RX	TX/RX	RX
NCAIU ACE-Lite	TX	TX	TX						TX/RX	TX/RX	RX
NCAIU ACE-Lite-E	TX/RX	TX	TX						TX/RX	TX/RX	RX
DCE											
DII	TX/RX	TX/RX			TX/RX	TX/RX	TX/RX				RX
DMI	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX	TX/RX				RX
DVE	TX	TX	TX	TX	TX	TX	TX		TX	TX	

TABLE 7 DN TX RX CONNECTIVITY MAP

CDTI Blocks

There are four blocks used in the networks of the CDTI:

- Switches
- Pipe Adapters
- Clock Adapters
- Width Adapters

Please refer to Legato specification for details on these components

2.2 CSTI Network

The CSTI network is used to access configuration, status and error registers in Ncore.. It is composed of two sub networks:

1. Request network
2. Response network

CSTI Blocks

In addition to the four blocks listed in the CDTI network, the CSTI network adds two more blocks

1. AXI initiator
2. APB target

The widths of all the CSTI blocks are fixed at 32 bits, which is a width that is not available in the CDTI network. Please refer to Legato specification for details on these components

3 Addressing and Memory Regions

Ncore address map is categorized into three main spaces:

- Ncore Register Space (NRS)
- General Purpose Address Space (GPAS)
- Boot Region (BR)
- Each space may contain one or more address regions.
  NRS is address space reserved for peripheral storage space, hosting Ncore unit configuration and status registers (CSR).
  The Organization of NRS is shown in Figure 2.

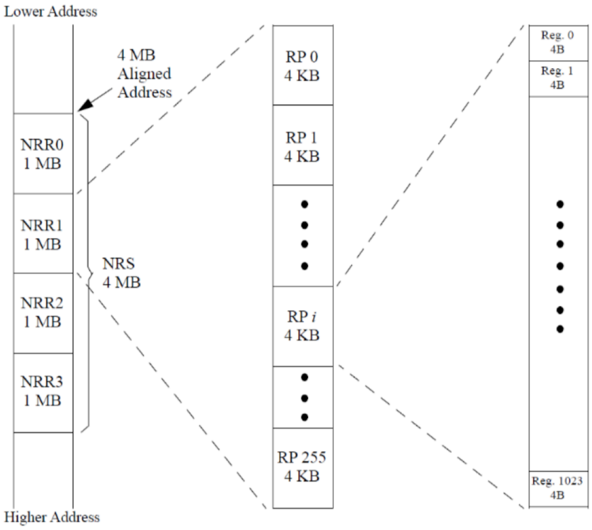


FIGURE 2: NRS STRUCTURE

3.1 Ncore Register Space

NRS may contain one or more Ncore register regions (NRR). Each NRR is of 1MB and is further divided into 256 register pages (RP) of 4KB.

Each Ncore component is associated with an RP and may implement up to 1024, 4 Byte (32-bit) registers.

Currently, Ncore system supports only one NRR. The number of valid RPs within this NRR is equal to the total number of Ncore components (Function Units, FU) configured in the system.

Table 8 shows the assignment of RPs to different Ncore components. Note that within the table

- nCAIUs refers to the number of CAIUs in the system, and the number of other Ncore components.
- nDII is always equal to total number of DIIs configured, plus one. The additional DII is the Ncore system DII that will be implicitly mapped into the NRS
- There will alway be only nDVE and nGRB

Item	Ncore Component	RPNs
1	CAIU (Coherent AIU)	RPN(0) to RPN(nCAIU - 1)
2	NCAIU (Non-Coherent AIU)	RPN0nCAIU) to RPN(nCAIU + nNCAIU - 1)
3	DCE (Distributed Cohrence Engine)	RPN(nCAIU + nNCAIU) to RPN(nCAIU + nNCAIU + nDCE - 1)
4	DMI (Distributed Memory Interface)	RPN(nCAIU + nNCAIU + nDCE) to RPN(nCAIU + nNCAIU + nDCE +nDMI - 1)
5	DII (Distributed I/O Interface)	RPN(nCAIU + nNCAIU + nDCE + nDMI) to RPN(nCAIU + nNCAIU + nDCE + nDMI + nDII - 1)
6	DVE (Distributed Virtualization Engine)	RPN(nCAIU + nNCAIU + nDCE + nDMI + nDII + 1)
7	GRB	0xFF (255)

TABLE 8: RP ASSIGNMENT TO NCORE COMPONENTS

The base address of this region shall be specified at Ncore system build time and must be 4MB aligned. Figure 7 shows the address distribution.

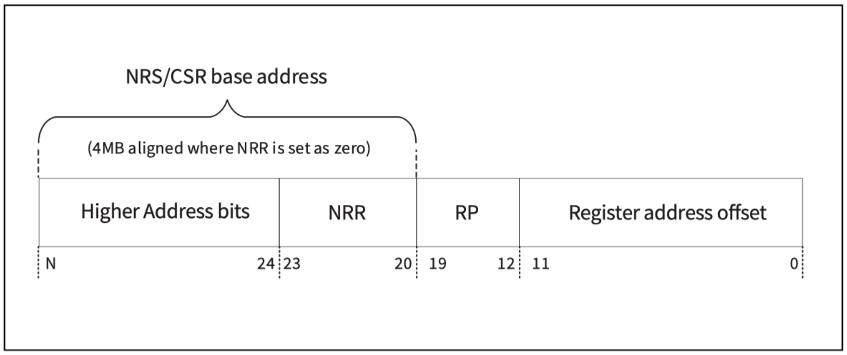


FIGURE 3: ADDRESS BIT MAPPING

3.2 General Purpose Address Space

General purpose address space is for general purpose use. It can support multiple number of address regions. The number of regions needed within a system must be specified at build time, minimum one region is required. The regions can only be configured via CSR accesses and must be configured by software at boot up.

They can be mapped to either normal memory (DMI) or a peripheral device (DII). The size of these regions must be a power of 2 within the range of 4 KB and 32 TB (achieved via interleaving). The base address configured must be aligned to the size of the region.

Boot address space is to be used for system bootup. Current Ncore system supports only one boot address region. This region can be mapped to either normal memory (DMI) or a peripheral device address space (DII). The size of this region must be a power of 2 within the range of 4 KB and 8 TB. The base address of this region must be specified at Ncore system build time and must be aligned to the size of the region.

Address decoding is priority based. NRS has the highest priority followed by general purpose address space and lastly boot address space. If an access matches higher priority address space, then overlapping lower priority address space matches are ignored, and the higher priority space is selected for access. Within an address space multiple address region matches are treated as errors; this specifically applies to general purpose address space. An access will result in an error if multiple address region matches are detected within a single address space or no matches are detected to any address space.

3.3 Interleaving

Ncore supports interleaving across snoop filters and system caches. This capability enables higher access bandwidth and eases physical design of the chip. Snoop filters are implemented within DCE and can be interleaved/distributed across 1 to 8 instances of DCE. System caches are implemented within DMI.

DMIs can be interleaved/distributed as a Memory Interleave Group (MIG) of 1, 2 or 4 DMIs. Multiple memory interleave groups form a Memory Interleave Group Set (MIGS). For example, if a system has 7 DMIs, DMI0 to DMI6, they can be

categorized into 3 different memory interleave groups:

- Group 0 of four DMIs (DMI0, DMI1, DMI2, DMI3);
- Group 1 of 2 DMIs (DMI4 and DMI5) and
- Group 2 of one DMI i.e. DMI6.

Three groups together are referred to as Memory Interleave Group Set. Alternatively, the MIGS may be configured as follows:

- Group 0 of two DMIs (DMI0, DMI1);
- Group 1 of two DMIs (DMI2, DMI3);
- Group 2 of two DMIs (DMI4 and DMI5) and
- Group 3 of one DMI i.e. DMI6

Examples for groups have been illustrated in Figure 4

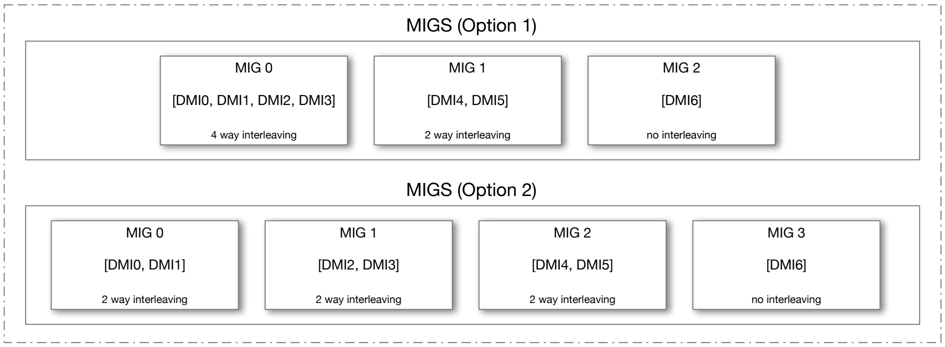


FIGURE 4 MIGS OPTIONS (EXAMPLES)

3.4 Instance and Set Selection

DCE & DMI component instances, system memory cache sets, proxy cache sets and snoop filter sets are selected as a function of address bits. The method for calculating the selection index is configurable, and the selection index is based on simple interleaving, in which the selection index is determined by the value of a set of N address bits, known as the primary selection bits. The value of selection index bit n equals the value of the address bit defined by primary selection bit n. For example, if primary selection bit 0 is specified as address bit 6, the value of selection index bit 0 equals the value of address bit 6.

Each address bit in the set of primary selection bits must be unique, i.e. an address bit must appear only once in the set of primary selection bits. The address bits must be chosen from a range whose upper bound is less than the size of the system address and whose lower bound is greater than or equal to the size of a cacheline. For example, if the size of the system address is 4 GB and the size of the directory cacheline is 64 Bytes, the highest address bit that can be selected is 31 and the lowest address bit is 6. Address bits less than the size of the directory cacheline comprise the directory cacheline offset, which are bits 0 through 5 in this example.

An identifier is computed by taking the value of the selection index modulo the number of items being selected. To select an instance on a component, an instance identifier is calculated. To select a set in a system memory cache, a proxy cache, or a snoop filter, a set identifier is calculated. In the case of Snoop filter and system memory cache the final

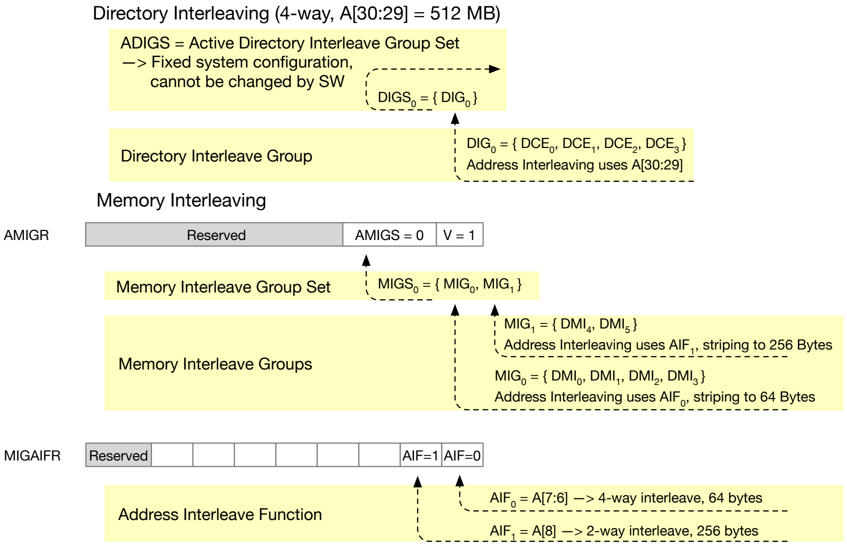

Directory Interleaving (4-way, A[30:29] = 512 MB)
ADIGS = Active Directory Interleave Group Set
—> Fixed system configuration,
cannot be changed by SW
DIGS$_0$ = { DIG$_0$ }
Directory Interleave Group
DIG$_0$ = { DCE$_0$, DCE$_1$, DCE$_2$, DCE$_3$ }
Address Interleaving uses A[30:29]
Memory Interleaving
AMIGR
Reserved
AMIGS = 0
V = 1
Memory Interleave Group Set
MIGS$_0$ = { MIG$_0$, MIG$_1$ }
Memory Interleave Groups
MIG$_1$ = { DMI$_4$, DMI$_5$ }
Address Interleaving uses AIF$_1$, striping to 256 Bytes
MIG$_0$ = { DMI$_0$, DMI$_1$, DMI$_2$, DMI$_3$ }
Address Interleaving uses AIF$_0$, striping to 64 Bytes
MIGAIFR
Reserved
AIF=1
AIF=0
Address Interleave Function
AIF$_0$ = A[7:6] —> 4-way interleave, 64 bytes
AIF$_1$ = A[8] —> 2-way interleave, 256 bytes

FIGURE 5: MEMORY AND DIRECTORY INTERLEAVING

set index is computed by concatenating the instance identifier with the set identifier, i.e. the instance identifier represents the most significant bits of the set index, while the set identifier represents the least significant bits. The address bits in the set of primary selection bits that determine the instance identifier must not appear in any of the sets of primary selection bits that determine the set identifier.

Additional constraints that apply to various components are described in more detail below.

Directory Interleaving

The coherency protocol in Ncore is directory based, directories are located within the DCE. To improve performance up to 16 directories may be interleaved.

- Interleaving is based on one (2-way interleaving), up to four address bits (16-way interleaving)
- Directories are mapped into groups (DIG = Directory Interleave Group), where each group uses a fixed interleave factor of 2^N (N = 0 .. 4), configured in **Maestro**
- Multiple Groups may be combined to form a DIGS
- DIGS (Directory Interleave Group Set), each set may only combine DIG in a way so that no DCE appears more than once in the merged list
- Multiple DIGS may be defined in **Maestro**, one of these must be selected at system start by configuring the ADIGR in each AIU

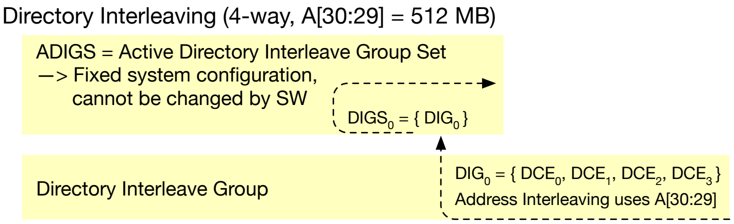

Directory Interleaving (4-way, A[30:29] = 512 MB)
ADIGS = Active Directory Interleave Group Set
—> Fixed system configuration, cannot be changed by SW
$DIGS_0 = \{ DIG_0 \}$
Directory Interleave Group
$DIG_0 = \{ DCE_0, DCE_1, DCE_2, DCE_3 \}$
Address Interleaving uses A[30:29]

FIGURE 6: DIRECTORY INTERLEAVING

Snoop Filter, System Memory Cache and Proxy Cache

The Set selection is additionally constrained as following:

- The number of sets per port must be a power-of-two integer
- The number of primary selection bits must equal the binary logarithm of the number of sets.

Distributed Memory Interface Selection

Ncore may be configured to use up to 16 Distributed Memory Interfaces (DMI). For performance, to balance the load and to distribute traffic evenly, DMI may be interleaved. The interleaving options are configured in **Maestro** by defining up to 8 Address Interleaving Functions.

- Interleaving is implemented by an Address Interleave Function (AIF)
- An AIF is based on 0 (no interleaving) to 4 (16-way) address bits and must be defined in **Maestro** (hard-coded). Each predefined AIF is assigned a unique 3-bit AIF ID.
- At least one AIF_m with a matching interleave factor N_m must exist for each defined MIG.
- DMI are mapped into Memory Interleave Groups (MIG). Each MIG must enumerate N members, where N is the interleaving factor. MIG are defined in **Maestro**.
- Ncore supports up to 8 MIG – an active AIF may be set for each MIG by setting the associated field in the MIGAIFR
- Multiple MIG may be combined to form a Memory Interleave Group Set (MIGS), each set may only combine MIG in a way so that no DCE appears more than once in the merged list
- Multiple MIGS may be defined in **Maestro**, one of them must be selected at system start by configuring the AMIGRs in each DCE and AIU

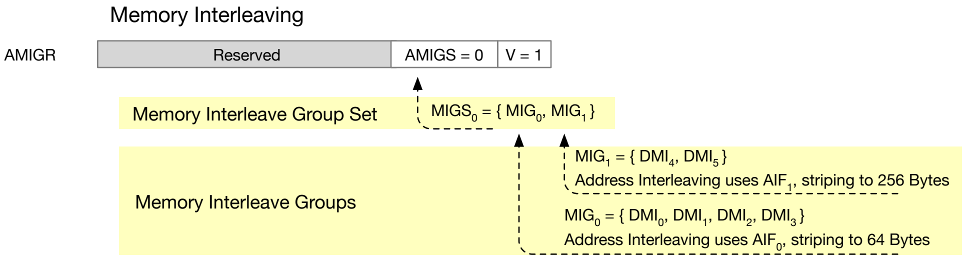

Memory Interleaving
AMIGR
Reserved
AMIGS = 0
V = 1
Memory Interleave Group Set
$MIGS_0 = \{ MIG_0, MIG_1 \}$
Memory Interleave Groups
$MIG_1 = \{ DMI_4, DMI_5 \}$
Address Interleaving uses AIF_1, striping to 256 Bytes
$MIG_0 = \{ DMI_0, DMI_1, DMI_2, DMI_3 \}$
Address Interleaving uses AIF_0, striping to 64 Bytes

FIGURE 7: MEMORY INTERLEAVING GROUP REGISTER

Distributed Coherency Engine Selection

DCE selection is additionally constrained as follows:

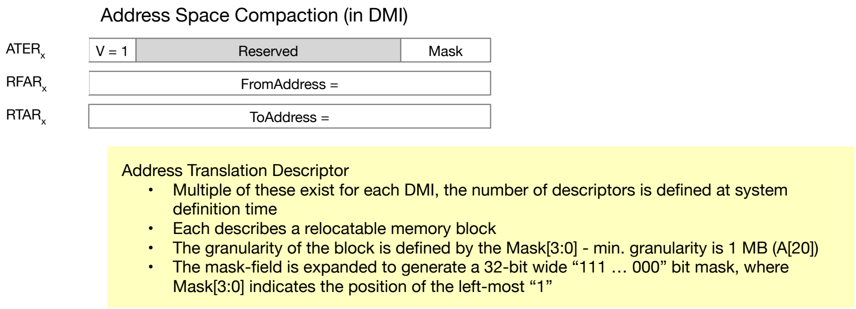
- The number of instances may be any integer within 1 to 8
- If the number of instances equals a power-of-two integer, the number of primary selection bits must equal the binary logarithm ($\log_2$) of the number of instances
- If the number of instances does not equal a power-of-two integer, the number of primary selection bits must equal an integer greater than the binary logarithm of the number of instances
- The primary selection bits must be chosen from the address bits common to all snoop filters and above the cacheline offset

If the number of ports is not equal to a power-of-two integer, the distribution of snoop filter cacheline addresses is non-uniform. For example, if the DCE is configured with three instances, the minimum number of primary selection bits equals two. In this configuration, 50% of cacheline addresses maps to one instance, and 25% maps to each of the other two instances. The non-uniformity may be smoothed by increasing the number of primary selection bits. By specifying four primary selection bits, 37.5% of cacheline addresses map to one instance, and 31.25% of cacheline addresses maps to each of the other two instances. By making the distribution of cacheline addresses more uniform, hot-spots are less likely to develop at an DCE instance.

Address Space Compaction

Through N-way interleaving, each memory region will be spread to N DMI modules – each memory request will arrive at DMI_x with an address relative to the base of the region and needs to be relocated to remove gaps in the physical memory address space

- Each DMI provides a configurable memory remapping function to support address space compaction
- The number of descriptors to describe compactable blocks is defined at configuration time in **Maestro**
- Each descriptor has 3 programmable registers
  - ATER - Address Translation Enable Register
  - RFAR - Relocation From Address Register
  - RTAR - Reolocation Target Register



Address Translation Descriptor

- Multiple of these exist for each DMI, the number of descriptors is defined at system definition time
- Each describes a relocatable memory block
- The granularity of the block is defined by the Mask[3:0] - min. granularity is 1 MB (A[20])
- The mask-field is expanded to generate a 32-bit wide “111 … 000” bit mask, where Mask[3:0] indicates the position of the left-most “1”

Figure 8: Address Space Compaction

The incoming physical memory address will be masked off by a 32-bit mask and compared to the RFAR (relocation "from" address register) in all valid ATR (Address Translation Descriptor)

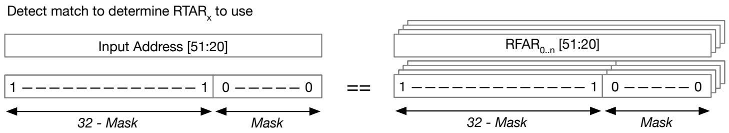


Figure 9: Matching Input address to ATR descriptor

If a descriptor matches, the masked value of the RTAR (relocation target address register) will replace the previously masked MSBs in the original address

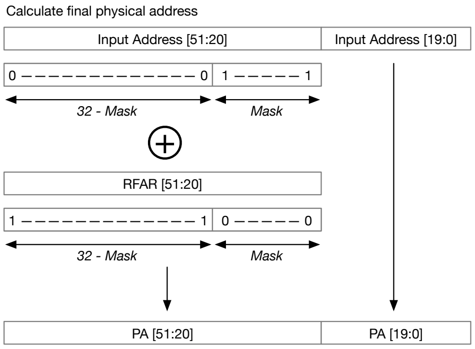


Figure 10: Relocation to new Base Address

Definition of a Memory Address Map

As previously described, General Purpose Address Space is partitioned into regions. Regions are defined by region descriptors in GPRegister space.

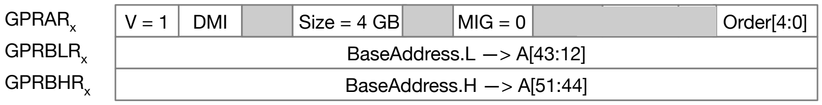


FIGURE 11: LAYOUT OF GP ADDRESS SPACE DESCRIPTOR

Each Agent-Interface (AIU) for Ncore 3 implements address space decoding; it decodes addresses of incoming memory transactions and directs these to the target servicing the addressed memory or I/O within the global address space.

- The number of descriptors in each unit must be configured in Maestro
- Each aperture is described by a set of 3 General Purpose Region Definition Registers
  - Each active descriptor (V == 1) configures:
  - The home unit type (DMI/DII) where this address space resides
  - Home-Unit-ID (HUI)
    - DII are enumerated, starting at 0 (DID)
    - For DMI the MIG for the region will be used
  - The BaseAddress and Size of the region:
    - Two registers are required to define the base address of a region
      - GPRBLR = AddrLow
      - GPRBHRx = AddrHigh
      - The base address of any aperture is aligned to 4096 bytes.
    - For coherent address space: a directory interleaving group. Directory interleaving groups are defined in **Maestro** , these are hard-coded at design time
  - One special descriptor for the Boot Region needs to be fully defined and will be pre-initialized in Maestro
  - Overlapping address space regions are decoded in priority order:
    - BootRegion < GPRegion < NRS (Ncore Register Space)

Bit	Name	Description	Access	Reset
31:0	AddrLow	Low order bits 43:12 of a region's base address	R/W	0x00000000

Bit	Name	Description	Access	Reset
7:0	AddrHigh	High order bits 51:44 of a region's base address	R/W	0x00
31:8	Rsvd	Reserved	RO	0x000000

TABLE 9: APERTURE BASE REGISTERS

The third register, GPRARx, defines the region attributes. The number of apertures is configurable by the user, a special (default) boot region exists and uses the same register format. A similar set of registers exists in DCE.

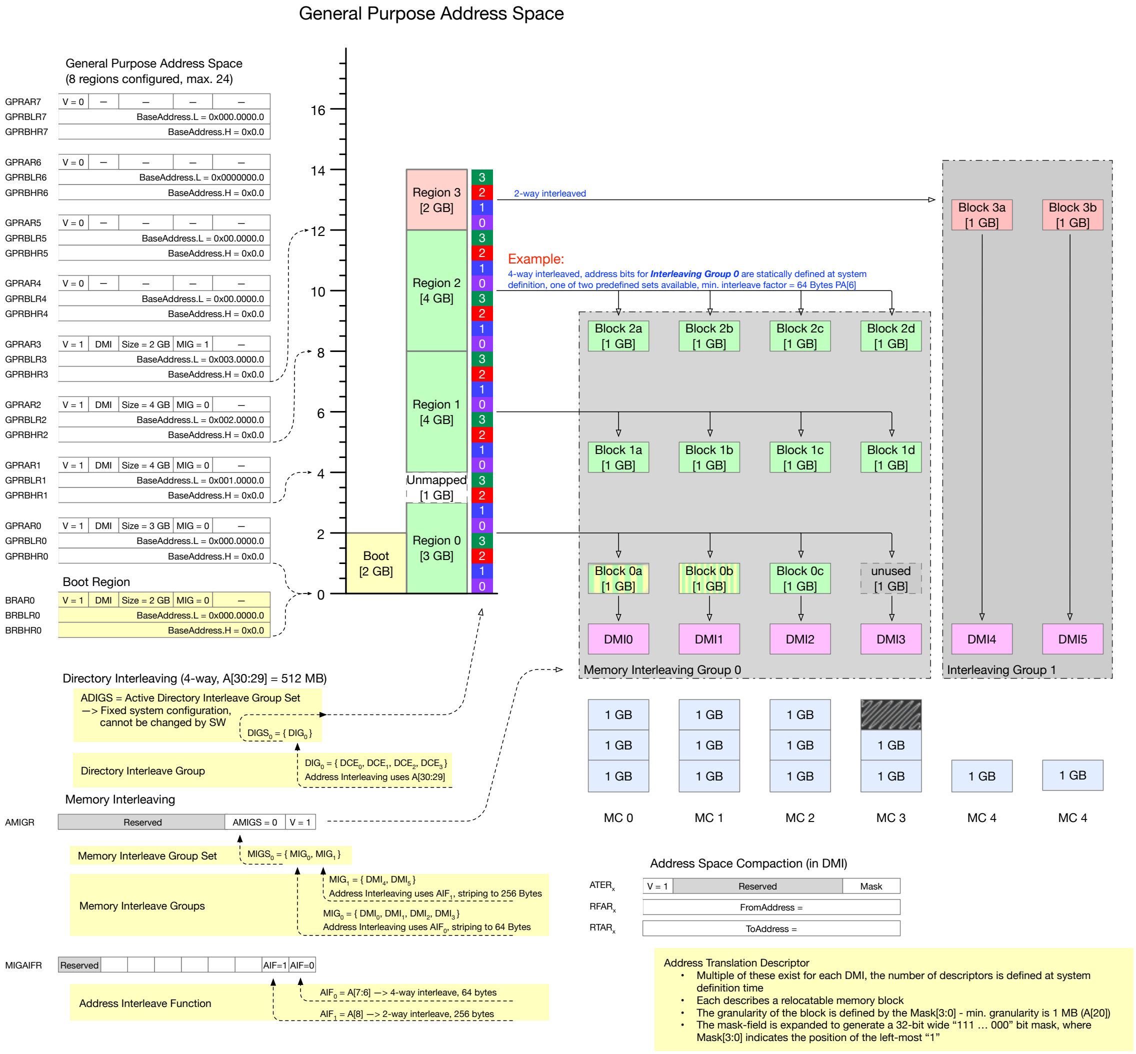

General Purpose Address Space
General Purpose Address Space
(8 regions configured, max. 24)
GPRAR7 | V = 0 | — | — | — | —
GPRBLR7 | BaseAddress.L = 0x000.0000.0
GPRBHR7 | BaseAddress.H = 0x0.0
GPRAR6 | V = 0 | — | — | — | —
GPRBLR6 | BaseAddress.L = 0x0000000.0
GPRBHR6 | BaseAddress.H = 0x0.0
GPRAR5 | V = 0 | — | — | — | —
GPRBLR5 | BaseAddress.L = 0x00.0000.0
GPRBHR5 | BaseAddress.H = 0x0.0
GPRAR4 | V = 0 | — | — | — | —
GPRBLR4 | BaseAddress.L = 0x00.0000.0
GPRBHR4 | BaseAddress.H = 0x0.0
GPRAR3 | V = 1 | DMI | Size = 2 GB | MIG = 1 | —
GPRBLR3 | BaseAddress.L = 0x003.0000.0
GPRBHR3 | BaseAddress.H = 0x0.0
GPRAR2 | V = 1 | DMI | Size = 4 GB | MIG = 0 | —
GPRBLR2 | BaseAddress.L = 0x002.0000.0
GPRBHR2 | BaseAddress.H = 0x0.0
GPRAR1 | V = 1 | DMI | Size = 4 GB | MIG = 0 | —
GPRBLR1 | BaseAddress.L = 0x001.0000.0
GPRBHR1 | BaseAddress.H = 0x0.0
GPRAR0 | V = 1 | DMI | Size = 3 GB | MIG = 0 | —
GPRBLR0 | BaseAddress.L = 0x000.0000.0
GPRBHR0 | BaseAddress.H = 0x0.0
Boot Region
BRAR0 | V = 1 | DMI | Size = 2 GB | MIG = 0 | —
BRBLR0 | BaseAddress.L = 0x000.0000.0
BRBHR0 | BaseAddress.H = 0x0.0
16
14
12
10
8
6
4
2
0
Boot [2 GB]
Region 0 [3 GB]
Unmapped [1 GB]
Region 1 [4 GB]
Region 2 [4 GB]
Region 3 [2 GB]
3 2 1 0
2-way interleaved
Example:
4-way interleaved, address bits for Interleaving Group 0 are statically defined at system definition, one of two predefined sets available, min. interleave factor = 64 Bytes PA[6]
Block 2a [1 GB]
Block 2b [1 GB]
Block 2c [1 GB]
Block 2d [1 GB]
Block 1a [1 GB]
Block 1b [1 GB]
Block 1c [1 GB]
Block 1d [1 GB]
Block 0a [1 GB]
Block 0b [1 GB]
Block 0c [1 GB]
unused [1 GB]
Block 3a [1 GB]
Block 3b [1 GB]
DMI0
DMI1
DMI2
DMI3
DMI4
DMI5
Memory Interleaving Group 0
Interleaving Group 1
1 GB
MC 0
MC 1
MC 2
MC 3
MC 4
MC 4
Directory Interleaving (4-way, A[30:29] = 512 MB)
ADIGS = Active Directory Interleave Group Set
—> Fixed system configuration, cannot be changed by SW
DIGS$_0$ = { DIG$_0$ }
Directory Interleave Group
DIG$_0$ = { DCE$_0$, DCE$_1$, DCE$_2$, DCE$_3$ }
Address Interleaving uses A[30:29]
Memory Interleaving
AMIGR | Reserved | AMIGS = 0 | V = 1
Memory Interleave Group Set
MIGS$_0$ = { MIG$_0$, MIG$_1$ }
Memory Interleave Groups
MIG$_1$ = { DMI$_4$, DMI$_5$ }
Address Interleaving uses AIF$_1$, striping to 256 Bytes
MIG$_0$ = { DMI$_0$, DMI$_1$, DMI$_2$, DMI$_3$ }
Address Interleaving uses AIF$_0$, striping to 64 Bytes
MIGAIFR | Reserved | AIF=1 | AIF=0
Address Interleave Function
AIF$_0$ = A[7:6] —> 4-way interleave, 64 bytes
AIF$_1$ = A[8] —> 2-way interleave, 256 bytes
Address Space Compaction (in DMI)
ATER$_x$ | V = 1 | Reserved | Mask
RFAR$_x$ | FromAddress =
RTAR$_x$ | ToAddress =
Address Translation Descriptor
• Multiple of these exist for each DMI, the number of descriptors is defined at system definition time
• Each describes a relocatable memory block
• The granularity of the block is defined by the Mask[3:0] - min. granularity is 1 MB (A[20])
• The mask-field is expanded to generate a 32-bit wide "111 … 000" bit mask, where Mask[3:0] indicates the position of the left-most "1"

FIGURE 12: EXAMPLE ADDRESS MAP

Bit	Name	Description	Access	Reset
0	Rsvd	Reserved	RO	0x0
1	ReadID (ARID)	No ordering by AxID (free listing) for reads[1]	RW	0x0
2	WriteID (AWID)	No ordering by AxID (free listing) for writes[15]	RW	0x0

[1] Setting ReadID and WriteID (freelisting) will force the internal OR[1:0] field to 2b00 (unordered) for memory targets

Bit	Name	Description	Access	Reset
4:3	Policy[1:0] 11 Endpoint Order 10 Relaxed Order 0X Reserved	Policy determines the setting of the internal OR[1:0] field which defines the behavior of the agent at the bottom of Ncore Endpoint[2] order: ▪ All transactions to the same peripheral region are issued and completed in order ▪ Override the ordering mode of the incoming transaction to *Endpoint Order* Relaxed order[3]: ▪ Responses within a flow (same AxID) are ***ordered*** if ReadID/WriteID bits are zero, otherwise they are ***unordered*** ▪ ***Ordered*** transactions will be issued as soon as the previous transaction in the ordered sequence has been ordered at the target (StrReq has been returned) and command credits are available ▪ ***Unordered*** transactions will be issued as soon as command credits are available at the target Write: BRESP may be returned (earliest) as soon as DtwRsp has been received, following AXI ordering rules (all older responses have been sent)	RW	0x0
8:5	Rsvd	Reserved **Note:** some of these bits have already assigned use in the next release of Ncore	RO	0x0
13:9	HUI	Home Unit Identifier used to access this memory regions - this field uses the index of the target HU within the enumerated set of DMI or DII	RW	0x0
19:14	Rsvd	Reserved	RO	0x0
24:20	Size	This field describes the size of the aperture in address space. The size of a region is calculated as: size_of(IG) * $2^{(Size + 12)}$ with IG = Interleave group	RW	
28:25	Rsvd	Reserved	RO	0x0
29	Rsvd	Reserved - this bit has been reserved to be used as an extension to HUT when hierarchical gateways or chip-to-chip interfaces will be implemented	RO	0x0
30	HUT	This field describes the Home Unit Type 0: System Memory 1: I/O (Peripheral Memory/Devices)	RW	0x0
31	Valid	This bit indicates if an aperture descriptor is valid 0: Invalid Mapping 1: Valid region descriptor	RW	0x0

Note: ReadID/WriteID/Policy - These fields are not applicable to CHI or ACE transactions and shall be reserved in CAIU

TABLE 10: APERTURE ATTRIBUTE REGISTER

[2] Read: Issue --> DtrReq, Response --> DtrReq --- Write: Issue --> DtwRsp, Response --> DtwRsp. Setting ReadID or WriteID bits is not recommended for this policy and may result in undefined behavior

[3] The equivalent of ***Strict Response*** ordering can be achieved by using ***Relaxed Order*** and setting the ReadID and WriteID bits

4 PCIe Compatibility

PCIe supports multiple interleaved transactions, sharing a common transport medium. The different transaction types are listed in table 1 (Transaction Type column). The PCIe root complex in an SoC would convert these transactions into ACE-Lite protocol when interfacing to an Ncore 3 IOAIU.

Depending on the address space selected and the semantics of the transaction, Ncore 3 would then propagate these to one of 3 types of functional interfaces:

- coherent memory --> DCE
- non-coherent memory --> DMI
- non-coherent memory and device transactions --> DII
- The PCIe IP ACE-Lite bus master will only use ReadNoSnoop, WriteNoSnoop, ReadOnce and WriteUnique transactions

Item	Transaction Type	Mode	Target	ACE-Lite Transaction - AxCACHE[3:0]
1	Memory Read	Non-Posted	DCE[4]	normal, cacheable, bufferable
2			DMI	normal[5]
3			DII	normal, non-cacheable[6]
4	Memory Write	Posted	DCE	normal, cacheable, bufferable
5			DMI	normal, non-cacheable, bufferable
6			DII	normal, non-cacheable, bufferable/non-bufferable[7]
7	Memory Read Lock[8]	Non-Posted	DCE/DMI	normal, non-bufferable
8			DII	normal, non-cacheable, non-bufferable
9	IO-Read	Non-Posted	DII	device, non-bufferable
10	IO-Write	Non-Posted	DII	device, non-bufferable
11	Configuration Read (Type 0/1)	Non-Posted	DII	device, non-bufferable
12	Configuration Write (Type 0/1)	Non-Posted	DII	device, non-bufferable
13	Message	Posted	DII	device, non-cacheable, non-bufferable
14	AtomicFetchAdd[9]	Non-Posted	DCE/DMI	normal, cacheable, non-bufferable
15	AtomicSwap[9]	Non-Posted	DCE/DMI	normal, cacheable, non-bufferable
16	AtomicCAS[9]	Non-Posted	DCE/DMI	normal, cacheable, non-bufferable

Table 11 Expected PCIe Transaction to AXI mapping

[4] All cacheable memory accesses are bufferable

[5] Technically any normal memory access is supported

[6] Cacheable transactions are not expected

[7] Both transaction types are legal - SNPS IP uses non-bufferable transactions

[8] AXI4 does no longer support locked transactions

[9] Our ACE-Lite implementation supports atomics but atomics must be cacheable and will not terminate properly if the addressed region is not backed up by SMC or if the region is not addressing scratch pad memory

ARM protocol specifies certain behavior - which, for certain sequences requires some attention to avoid deadlocks, especially if traffic arriving from one PCIe device will be forwarded to a different PCIe device (bridge functionality).

Response:

- What AXI/ACE-Lite transaction encoding/attribute setting are MSI transactions using
  a. MSI Transactions are "Memory" type transactions - an MSI Write would be a "Memory Write"
  b. The bufferability attribute is generated (in default condition) based on the NS bit on the PCI link packets.

  If NS[10] = 1, ACELite = Non-Bufferable

  if NS = 0, ACELite = Bufferable

- What encoding/attribute setting distinguishes posted from non-posted memory writes

  Posted PCIE link Writes are always generated with AXI-ID = 0

  Non-Posted PCIE link Writes are always generated with AXI-ID!= 0

4.1 ACE-Lite Transactions generated by Synopsys PCIe Controller with AMBA Bridge

The controller supports the AXI4 protocol with additional support for the ACE-Lite protocol. To enable these features, you must set AMBA_INTERFACE =3 and CC_ACELITE_ENABLE =1.

ACE-Lite Features and Limitations:

- The AXI bridge slave does not implement ACE; it ignores the value of the DOMAIN, BAR and SNOOP fields; barrier transactions are not supported
- The AXI bridge master interface generates requests that are dependent upon the NS and TH bits
- The master interface always sets AxBAR and AxSNOOP to ‘0’ so that requests are restricted to ReadNoSnoop, WriteNoSnoop, ReadOnce, and WriteUnique
- You can override the value of the DOMAIN field, except for message requests where it is always 11
- The master does not support barriers
- Cache coherency is not optional, and the master always uses the NS bit, unless you override the DOMAIN signals

[10] NS = non-snoop property of PCIe TLP

Cache Coherency	Request Type	Memory Type[11] 0: Peripheral 1: Main Memory	NS[12]	TH	AxSNOOP	AxBAR	AxDomain	Shareability Domain	Memory Type
Non-coherent	Message	X	X	X	4b0000	2b00	2b11	System	Device, Non-bufferable
Non-coherent	Read or Write	0	X	X	4b0000	2b00	2b11	System	Device, Non-bufferable
Non-coherent	Read or Write	1	1	X	4b0000	2b00	2b11	System	Normal, Non-cacheable Non-bufferable
Coherent	Read	1	0	0	4b0000	2b00	2b10	Outer Shareable	Writeback, No-allocate
Coherent	Write	1	0	0	4b0000	2b00	2b10	Outer Shareable	Writeback, No-allocate
Coherent	Read	1	0	1	4b0000	2b00	2b10	Outer Shareable	Writeback, Read-allocate
Coherent	Write	1	0	1	4b0000	2b00	2b10	Outer Shareable	Writeback, Write-allocate

Table 12 ACE Lite Transactions - SNPS PCIe IP

The rules for PCIe transaction

For more details, see PCIe specification, Section 2.4, Transaction Ordering. **Error! Reference source not found.** defines the ordering requirements for PCIe Transactions. the columns represent a first issued transaction and the rows represent a subsequently issued transaction. The table entry indicates the ordering relationship between the two transactions. The table entries are defined as follows:

Yes The second transaction (row) must be allowed to pass the first (column) to avoid deadlock. (When blocking occurs, the second transaction is required to pass the first transaction. Fairness must be comprehended to prevent starvation.)

Y/N There are no requirements. The second transaction may optionally pass the first transaction or be blocked by it.

No The second transaction must not be allowed to pass the first transaction. This is required to support the producer/ consumer strong ordering model.

Row Pass Column?	Posted Request (Col 2)	Non-Posted Request	Completion (Col 5)

[11] Set by the CFG_MEMTYPE_VALUE field of the COHERENCY_CONTROL_1_OFF register

[12] NS = non-snoop property of PCIe TLP

			Read Request (Col 3)	NPR with Data (Col 4)	
Posted Request (Row A)		a) No b) Y/N	Yes	Yes Yes	a) Y/N b) Yes
Non-Posted Request	Read Request (Row B)	a) No b) Y/N	Y/N	Y/N	Y/N
	NPR with Data (Row C)	a) No b) Y/N	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

Table 13 PCIe Ordering Rules Summary

Posted Request is a Memory Write Request or a Message Request.

Non-Posted Request (with Data) Configuration Write Request, an I/O Write Request, or an AtomicOp Request.

Non-Posted Request Read Request or an NPR with Data

Row A, B, C

Special cases:

a) applies to all transactions unless condition b) applies

b) Requests are allowed to pass when

RO (relaxed ordering attribute) is set

IDO (ID ordering) applies and the Requester IDs are different

Both requests have been issued using different PASID

Row D 2a) Completion must not pass a Posted Request

2b) Exception:

An I/O or Configuration Write Completion is permitted to pass a Posted Request

Completions with IDO are permitted to pass a Posted Request if they belong to independent flows (C.ID <> R.ID)

5a) Completions with different Transaction IDs are permitted to pass each other

5b) Completions with the same Transaction ID are not allowed to pass each other

The rules for AMBA transactions

As a reference, see Arm's document IHI 0022G, Section A6.3 Transactions and Ordering.

A transaction is a read or a write to one or more address locations. The locations are determined by **AxADDR** and any relevant qualifiers such as the Non-secure bit in **AxPROT**.

- Ordering guarantees are given only between accesses to the same Memory location or Peripheral region when the same AxID is used.
- A transaction to a Peripheral region must be entirely contained within that region.
- A transaction that spans multiple Memory locations has multiple ordering guarantees.

Transactions can be either of type Device or Normal:

Device: Device transactions can be used to access Peripheral regions or Memory locations. In Ncore 3.0 architecture, these transactions will be forwarded to DII.

Normal: Normal transactions are used to access Memory locations and are not expected to be used to access Peripheral regions.

A Normal access to a Peripheral region must complete in a protocol-compliant manner, but the result is IMPLEMENTATION DEFINED. A write transaction can be either Non-Bufferable or Bufferable. It is possible to send an early response to Bufferable writes.

4.1.1.1 Guarantees before a completion response is received

In AMBA AXI/ACE-Lite protocol the transaction ID identifies a "flow" of associated transactions, originating at a requester agent and terminating at a target agent. Since the protocol does not identify individual transactions, ordering rules for transport, execution and responses are required to maintain consistency and to associate the request with a response.

All of the following guarantees apply to transactions from the same master, using the same ID:

- A Device write DW1 is guaranteed to arrive at the destination before Device write DW2, where DW2 is issued after DW1 and to the same Peripheral region.
- A Device read DR1 is guaranteed to arrive at the destination before Device read DR2, where DR2 is issued after DR1 and to the same Peripheral region.
- A write W1 is guaranteed to be observed by a write W2, where W2 is issued after W1 and to the same Memory location.
- A write W1 that has been observed by a read R2 is guaranteed to be observed by a read R3, where R3 is issued after R2 and to the same Memory location.

  The guarantees imply that there are ordering guarantees between Device and Normal accesses to the same Memory location.

4.1.1.2 Guarantees from a completion response

A completion response guarantees all of the following:

- A completion response to a read request guarantees that it is observable to a subsequent read or write request from any master.
- A completion response to a write request guarantees that it is observable to a subsequent read or write request from any master. This observability is a requirement of a system that is multi-copy atomic.

  Systems that contain Arm Architecture-compliant processors must be multi-copy atomic. That is, the Multi_Copy_Atomicity property must be ***True***.

The response to a Bufferable write request can be sent from an intermediate point. It does not guarantee that the write has completed at the endpoint, but it is observable to future transactions.

4.1.1.3 Response ordering guarantees

Transaction responses have all the following ordering guarantees:

- A read R1 is guaranteed to receive a response before the response to a read R2, where R2 is issued from the same master after R1 with the same ID.
- A write W1 is guaranteed to receive a response before the response to a write W2, where W2 is issued from the same master after W1 with the same ID.

What rules do we follow in IOAIU?

4.1.1.4 Current uArch specification

- Maintain separate AxID linked lists for read and writes, responses for same AxID will be in order within a read or write channel
- Generate Ordering bits based on AxCache[3:1] values for transactions with same AxID
  - 0: WriteNonCohFull - transaction with no older dependent transaction with same AxID (separate list for reads and writes)
  - 2: if (AxCache[3:1] > 0) (Normal) and there is an older transaction with same AxID (separate list for reads and writes)
  - 3: if (AxCache[3:1]==0) (Device) and there is an older transaction with same AxID (separate list for reads and writes)
- Add a CSR bit to run in producer consumer mode
  - In this mode IO AIU will block same AxID transactions until the STR response is sent for the dependent transaction (separate list for reads and writes)
  - When mode is not enabled then blocking is not required
- Same cache line address blocking until STR response is sent. This is done across reads and writes to avoid hazards (WAW, WAR, RAW)
- Writes will have at-least one or more reserved OTT entries and always make forward progress form IO AIU perspective.

The impact of the uArchitectural rules applied to PCIe transaction sequences

- The current implementation of Ncore does not support the concept of posted requests - order is enforced between all writes with the same AxID
  - 'non-posted' writes will not pass 'posted' writes (C2a) - **required**
  - 'non-posted' writes will not pass 'non-posted' writes (C4) - **less optimal, but legal**[13]
  - 'posted' writes will not pass other 'posted' writes (A2a) with the same ID or address - **required**
  - 'posted' writes will not be able to pass 'non-posted' writes (A4a) with the same ID - in principle this violates the requirement but:
    a. If posted writes use a different ID than NPRs, this problem goes away[14]
    b. Ncore guarantees forward progress for writes vs. reads - posted writes will make forward progress[15]

[13] This are AXI rules - we allow to relax them for memory targets by address region (GPRAR registers)

[14] Strict request ordering based on the AxID - but performance suffers

[15] Yes, a single open entry does not look like much but, writes will always have priority over reads, so when OTT is full, a write will get in for every returning transaction (pipelining)

c. Is having no posted writes a problem with performance? - **Yes**

- relaxed ordering is implied by not using the same ID (A2b, A4b, C2a, C4) - **allowed**

- Read requests will pass all write requests, except if a hazard exists
  - read requests must not pass 'posted' write requests with the same ID and the same address[16] (B2a) - **implemented** (we respect WAR, RAW, WAW order, independent of AxID)
  - read requests may pass 'posted' write requests with a different ID (B2b) - **allowed**

Notes:

1. This are AXI rules - we allow to relax them for memory targets by address region (GPRAR registers)
2. Strict request ordering based on the AxID - but performance suffers
3. Yes, a single open entry does not look like much but, writes will always have priority over reads, so when OTT is full, a write will get in for every returning transaction (pipelining)
4. This is not explicitly stated, but inferred from memory consistency and R-W ordering requirements - reads will always return the data written by a write that has been issued earlier.
5.

Performance improvement by relaxed ordering rules

- We define ordering by address region (CMN does it that way) instead of target FUnit; each memory region is defined by a GPRAR. A new order control field has been added in each GPRAR$_x$

<table>
<tr><td>GPRAR$_x$</td><td>V = 1</td><td>DMI</td><td></td><td>Size = 4 GB</td><td></td><td>MIG = 0</td><td></td><td>NS[1:0]</td><td>NC</td><td>Order[4:0]</td></tr>
<tr><td>GPRBLR$_x$</td><td colspan="10">BaseAddress.L —> A[43:12]</td></tr>
<tr><td>GPRBHR$_x$</td><td colspan="10">BaseAddress.H —> A[51:44]</td></tr>
</table>

- The control field (bits 0.. 2) within this register controls the rules when a transaction may be **issued** to the internal target (DCE/DMI or DII)[17]
- The **Policy** field determines the behavior at the target agent - DCE and DMI ignore the policy setting - and the response behavior
- Device transactions (AxCACHE[3:1] == 3b000) will enforce strict request ordering (Policy == 2b11) for all requests with the same AxID - these shall only be used for targets behind a DII.
- Memory request addressing DII targets may use stricter rules (Request order)

Order	Function	Description
0	Reserved	Always read as zero
1	ReadID (ARID)	No ordering by AxID (free listing) for reads[18]

[16] This is not explicitly stated, but inferred from memory consistency and R-W ordering requirements - reads will always return the data written by a write that has been issued earlier.

[17] Read/Write ordering rules may be different for each target region - Setting ReadID/WriteID field overrides AxID ordering!

[18] Setting ReadID and WriteID (freelisting) will force the internal OR[1:0] field to 2b00 (unordered) for memory targets

2	WriteID (AWID)	No ordering by AxID (free listing) for writes[15]
4:3	Policy 11 Endpoint Order 10 Relaxed Order 01 Reserved 00 Reserved	Policy determines the setting of our internal OR[1:0] field which defines the behavior of the agent at the bottom of Ncore Endpoint[19] order: ▪ All transactions to the same peripheral region are issued and completed in order ▪ Override the ordering mode of the incoming transaction to *Endpoint Order* Relaxed order[20]: ▪ Responses within a flow (same AxID) are ***ordered*** if ReadID/WriteID bits are zero, otherwise they are ***unordered*** ▪ ***Ordered*** transactions will be issued as soon as the previous transaction in the ordered sequence has been ordered at the target (StrReq has been returned) and command credits are available ▪ ***Unordered*** transactions will be issued as soon as command credits are available at the target ▪ Write: BRESP may be returned (earliest) as soon as DtwRsp has been received, following AXI ordering rules (all older responses have been sent)

TABLE 14 CSR CONFIGURATION

Recommended Settings

AxCACHE[3:0]	Target	Policy [1:0]	WriteID (AWID)	ReadID (ARID)	Description
4b000x	DII - Device	2bxx	0	0	Device property of transaction overrides policy setting
4bxx1x	DII - Memory	2b11	0	0	Transaction to memory mapped device functions (GIC) - this setting guarantees completion of all outstanding writes to any target using the same AWID
4bxx1x	DII - Memory	2b10	0	0	Memory transaction (SRAM, ROM, Flash) - AxID ordered requests, Hazard check at PoS[21] or in AIU
4bxx1x	DII - Memory	2b10	0	1	Memory devices with write ordering requirements (some Flash), Hazard check in AIU
4bxx1x	single DCE/DMI	2b10	1	1	Issue of transactions free listed, Response follows Relaxed Order policy, Hazard check in AIU Best bandwidth
4bxx1x	interleaved DCE/DMI	2b10	1	1	Issue of transactions free listed, Response follows Relaxed Order policy, Hazard check in AIU Best bandwidth
4bxx1x	DCE/DMI	2b10	0	0	AXI ordering enforced - severe bandwidth penalty

TABLE 15 RECOMMENDED SETTINGS

[19] Read: Issue --> DtrReq, Response --> DtrReq --- Write: Issue --> DtwRsp, Response --> DtwRsp. Setting ReadID or WriteID bits is not recommended for this policy and may result in undefined behavior

[20] The equivalent of ***Strict Response*** ordering can be achieved by using ***Relaxed Order*** and setting the ReadID and WriteID bits

[21] Khaleel says, that hazard check may fail under rare circumstances - need to understand the conditions and fix if necessary

Concerns about complete "freewheeling"

Sequential Consistency[22] cannot be guaranteed for interleaved memory transactions if multiple memory interfaces are interleaved - a sequence of memory writes by a writer will alternately/sequentially target different memory agents (e.g. interleaving 4 DCEs at 64 bytes will result in transactions for sequential cache lines to be sent to different DCEs and/or DMI). This may be no issue for most applications.

0x0000 --> DCE$_{00}$, 0x0040 --> DCE$_{01}$, 0x0080 --> DCE$_{02}$, 0x00c0 --> DCE$_{03}$, 0x0100 --> DCE$_{00}$ ... each DCE implements the Point-of-Coherence for part of the interleaved address space. If an observer agent's read requests do not arrive at the DCEs in the same order (due to different transport latency) the write-read order at each PoC may differ and writes may be observed in a different order than they were issued.

	DCE$_{00}$	DCE$_{01}$	DCE$_{02}$	DCE$_{03}$
t$_0$	W$_0$	W$_1$	W$_2$	W$_3$
t$_1$	R$_0$	R$_1$	W$_6$	R$_3$
t$_2$	R$_4$	R$_5$	R$_2$	R$_7$
t$_3$	W$_4$	W$_5$	R$_6$	W$_7$
t$_4$	R$_8$	R$_9$	W$_9$	W$_{10}$

Race condition between W$_6$ and R$_2$ has W$_6$ arrive earlier than R$_2$ - agent DCE$_{02}$ sees a different request order and the data returned in R$_3$ will be different (returning the previous content of memory location 2) than expected under the assumption of a sequential write. This is a problem of having multiple PoC processing a sequential data flow originating at a single source.

Streaming write ordering

Writes arriving at the IOAIU from an external agent (e.g. PCIe complex) follow AXI protocol. Write transactions are tagged with the AWID, all transactions carrying the same ID are part of the same flow and for memory targets shall be considered sequentially ordered within that flow.

In most use cases, the agent observing a sequential stream of writes within a flow needs to observe these writes exactly in the same order they were created. This is called Ordered Write Observation and required by sequential memory consistency.

Ncore implements Ordered Write Observation by making writes visible to observers when the ordering point receives the StrRsp message; this is the idication, that the write has been completed at the POS. In CCMP StrRsp concludes the write operation.

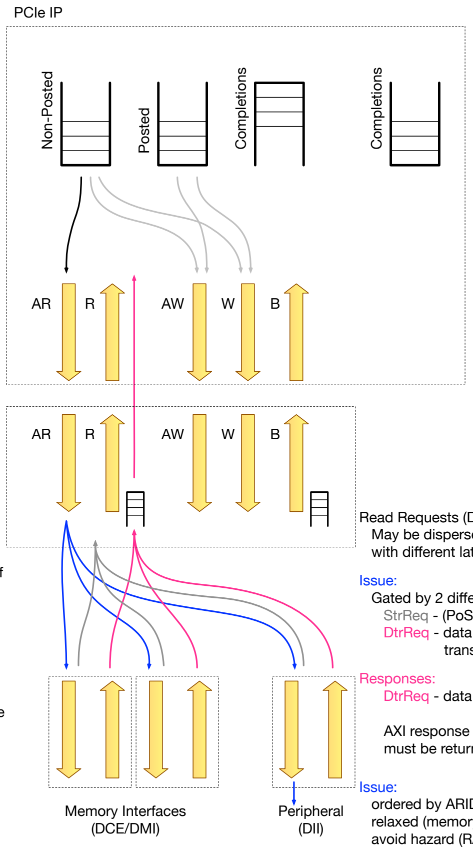
Transaction Progress through the system

4.1.1.5 *Acceptance of transactions*

In AXI or ACE-Lite protocol read and write transactions may be arriving concurrently. The protocol does not infer any ordering between read and write channels in that case. AXI or ACE-Lite transactions arriving at the NCAIU carry

[22] This is a standard problem when traffic is split to interleaved memory channels - this is a trade-off with performance impact. Applications using Producer-Consumer models need to be aware of this behavior and shall apply appropriate caution

AxCACHE[3:0] attributes - NCAIU shall treat any transaction with AxCACHE[3:1] == 3b000 as device ordered and enforce request order - these transactions are usually sent towards physical devices and will require strong ordering.



Read Requests:
memory - use unlisted ARID for best performance
device - enforce ordered issue to DII

Read Requests (DCE/DMI):
Will be dispersed to different agents if interleaved DCE/DMI, possibly finish with different latencies

Issue:
Gated by internal events:
StrReq - ordering (PoC/PoS)
DtrReq - data returned from source
Freelist - issue when credit available

Read Requests (DII):
May be dispersed to different agents, possibly finish with with different latencies

Issue:
Gated by 2 different internal events:
StrReq - (PoS) per ordering region (relaxed)
DtrReq - data returned through reorder buffer, transaction completed at target

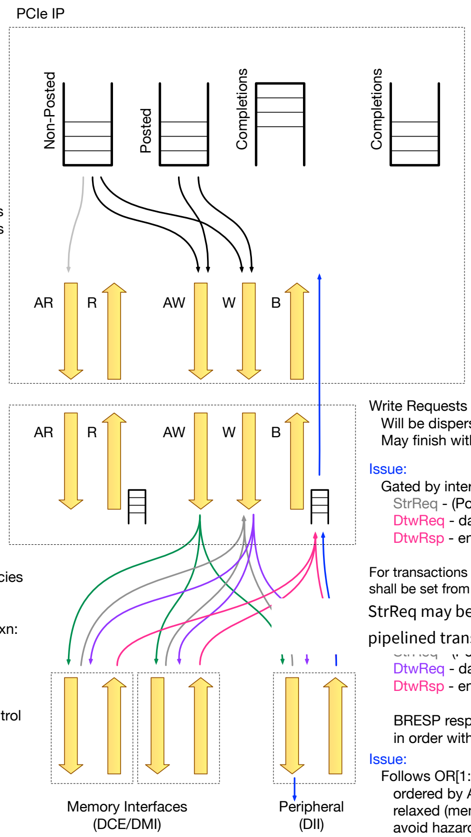
Responses:
DtrReq - data returned through reorder buffer (earliest)

AXI response for read depends on DtrReq and must be returned in order within each flow (ARID)

Issue:
ordered by ARID (device)
relaxed (memory)
avoid hazard (RAW, WAR, WAW)

Arm implements hazard check at (programmable) larger granularity than cache line — this achieves ordering between reads and writes within address region

FIGURE 13: TRANSACTION PROGRESS (READ)



Write Requests:
memory - use AWID = 0 for posted writes
use AWID != for all other writes
device - enforce ordered issue to DII

Write Requests (DII):
Will be dispersed to different agents
May finish with different latencies

Issue:
Gated by internal events, depends on CSR setting:
StrReq - (PoS) per ordering region (relaxed)
DtwReq - data sent towards target
DtwRsp - endpoint has accepted data

For transactions with AWCACHE Properties of OR[1:0] in transaction shall be set from control register configuration

StrReq may be used to send the next

pipelined transaction, ordering will occurr at y

StrReq - (PoS) per ordering region (relaxed)
DtwReq - data sent from reorder buffer
DtwRsp - endpoint has accepted data

BRESP response follows AXI rules and must be returned in order within each flow (AWID)

Issue:
Follows OR[1:0] ordering rule in transaction
ordered by AWID (device)
relaxed (memory)
avoid hazard (RAW, WAR, WAW)

Arm implements hazard check at (programmable) larger granularity than cache line — this achieves ordering between reads and writes within address region

Write Requests (DCE/DMI):
Will be dispersed to different agents
May become ordered with different latencies

Issue:
Gated by ordering events of dependent txn:
StrReq - ordering (PoC/PoS)
DtwReq - data sent to target
Freelist - issue when credit available

OR[1:0] in transaction may be set from control register configuration but will be ignored at destination.

Responses (DCE/DMI):

StrRsp is generated to make writes visible to other agents - controlled by

Order Policy	Description
00	unordered, StrRsp generated as soon as DtwRsp received from target
01 .. 11	Ordered, StrRsp generated when no older transaction with the same ID pending

GPRAR details

The GPRAR format has been modified from Ncore 3.0 to support new functionality, described in the previous sections of this document. The GPRARs add 5 bits to control the transaction behavior for each address aperture. The configuration bits 4:0 are currently not used across all AIU architectures and their function in CAIU will not change.[23]

CAIU GPRAR$_x$: General Purpose Region Attribute Register [Offset: 0x0400]				
Bit	**Name**	**Description**	**Access**	**Reset**

[23] CHI and ACE CAIU will not support these settings and propagate the protocol inherited attributes - register settings will be one either ignored or required to use a specified pattern (otherwise expect undefined behavior)

0	Rsvd	Reserved	RZWI	0x0
1	ReadID	No ordering by ARID (free listing)	RW	0x0
2	WriteID	No ordering by AWID (free listing)	RW	0x0
4:3	Policy	Response policy: 11: Endpoint/Request Order - do not set ReadID/WriteID! All memory transactions are issued and completed in order (issue from DtrReq/DtwRsp), BRESP from DtwRsp 10: Relaxed Order - next dependent transaction is issued as soon as ordering at the target has been achieved (StrReq), BRESP from DtwReq 01/00: Reserved	RW	0x0
5	NC	Non-Coherent 1: Non-coherent, any transactions in this aperture will be issued as non-coherent transactions 0: Coherent, any transactions issued here will be directed through the DCE and processed as coherent requests (IO coherent)	RZWI	0x0
7:6	NS[1:0]	Non-Secure - This is a 2-bit wide field, at this time NS[1] is reserved. x1: Non-secure, all transactions will be accepted x0: Secure, only transactions with property NS = 0 will be allowed, all other transactions will be terminated with an error response	RW	0x1
8	Rsvd	Reserved	RZWI	0x0
13:9	HUI	Identifier of the home unit, depending on type **Peripheral:** Index of target DII in enumerated list **Memory:** Memory Interleave Group **HierGateway:** Gateway Interleave Group (Reserved in 3.2) **Chip-to-Chip:** Gateway Interleave Group (Reserved in 3.2)	RW	0x0
19:14	Rsvd	Reserved	RZWI	0x0
24:20	Size	This field represents the size of the region in a binary number with a range of 0 .. 31. The size of the region is defined by RegionSize = IGSize * $2^{(size+12)}$ Bytes	RW	0x0
28:25	Rsvd	Reserved	RZWI	0x0
30:29	HUT	This field indicates the Home Unit Type: 00: System Memory 01: Hierarchical Gateway Interface (Not supported in 3.2) 10: Peripheral 11: Chip-to-Chip interface (Not supported in 3.2) Note, the pattern has been chosen to retain compatibility between Ncore 3.2 and future implementations. Ncore 3.2 implements only a single bit selector (bit 30) 0: System Memory 1: DII	RW	0x0
31	Valid	This bit indicates if the region is valid 0: Invalid mapping 1: Valid region mapping	RW	0x0

New Rules for the transaction processing

4.1.1.6 Transaction management

- Maintain separate linked lists for read and writes by AxID
  - transactions may be issued depending on ordering requirements using the AxID

        - AxID ordering when issueing requests towards the target agent may be disabled for an address region by GPRAR[1][24] for read accesses and GPRAR[2][25] for write accesses
            - Setting GPRAR[x] == 1 disables checking
            - The default value (after reset) will be 0 (checking enabled)
        - Responses for same AxID must be in order within a read or write channel - responses for individual flows (all transactions with the same AxID) from downstream agents may arrive out-of-order and must be reordered to meet AXI specification
    - Same cache line address blocking until the StrRsp has been sent. This is done across reads and writes to avoid hazards (WAW, WAR, RAW)
        - This check shall be controlled by GPRAR[0] the "hazard" bit
        - Checking shall be disabled when hazard == 1
        - The default value (after reset) will be 0 (checking enabled)

[24] Ignore ARID when set
[25] Ignore AWID when set

4.1.1.7 Transaction Issue

- Each transaction will be started by sending a CmdReq. The command includes a 2-bit wide "ordering" field OR[1:0]. This field determines request processing at the target agent (DCE, DMI, DII). Valid settings for OR[1:0] are

OR[1:0]	Function	Description
2b11	Endpoint ordered	Strongly ordered with respect to the same endpoint device
2b10	Strongly ordered[26]	Strongly ordered with respect to a previous access from the same source to the same location. Writes to the same address issued by the same agent (FUnitID and AxID) must meet requirements of sequential consistency
2b01	Reserved	
2b00	Not ordered	No ordering required for this access

- Target agents with memory semantics (DCE/DMI) will ignore the OR[1:0] setting
- Read/Write requests shall be issued when they are ready-to-issue, i.e. they meet certain conditions. The conditions to issue a transaction depend on the type of transaction (Read/Write), GPRAR [4:0], the AxCACHE[3:1] value and an "*Event*" trigger that removes the transaction from the dependency chain (eg. StrRsp). In AXI4, all transactions with AxCACHE[3:1] == 3b000 are considered "device" transactions and must be endpoint ordered

Type	Event	GPRAR[4:0]	OR[1:0]	Description
Read	DtrReq/StrReq	5bxxxx0	2b11	Endpoint ordered, transactions are issued in order[27]
Write	StrReq	5bxxxx0	2b11	

Table 16 Device Transactions (AxCACHE[3:0] = 4b000x)

Event[28]	GPRAR[4:0]	OR[1:0]	Description
StrReq/DtrReq[29]	5b10x00	2b10	Issue respects ARID as dependency, hazard check enabled
StrReq/DtrReq[29]	5b11xx0	2b11	Endpoint ordered, transactions are issued in order
---Error! Bookmark not defined.	5b10x10	2b00	Issue: Ignore ARID as dependency, no hazard detected, this transaction does not generate a dependency for subsequent transactions
StrReq/DtrReq[29,30]	5b10x10	2b00	Issue: Ignores ARID as dependency, hazard detected, this transaction does not generate a dependency for subsequent transactions

Table 17 Memory Read Transactions (AxCACHE[3:0] = 4bxx1x)

[26] Current documentation: ***Request Ordered***

[27] Initiator will not send another request until the current transaction has received RRESP/BRESP - therefore StrReq may be used to issue the next transaction to simplify the implementation

[28] Event refers to message that removes the transaction as a dependency

[29] Whatever event arrives first

[30] Ignoring AxID, transaction can progress when the transaction causing the hazard has been ordered

Event	GPRAR[4:0]	OR[1:0]	Description
StrReq	5b100x0	2b10	Issue respects AWID as dependency, hazard check enabled
StrReq	5b11xx0	2b11	Endpoint ordered, transactions are issued in order - a DII agent will return DtwRsp only after receiving BRSP
---Error! Bookmark not defined.	5b101x0	2b00	Issue: Ignore AWID as dependency, no hazard detected, this transaction does not generate a dependency for subsequent transactions
StrReq[30]	5b101x0	2b00	Issue: Ignores AWID as dependency, hazard detected, this transaction does not generate a dependency for subsequent transactions

Table 18 Memory Write Transactions (AxCACHE[3:0] = 4bxx1x)

4.1.1.8 Ordered Write Implementation

Ordered write transactions guarantee that sequential write requests arriving on one agent would become visible to any (different) observing agent in the same order they are issued. On the CHI interface this property will be enabled by hazard checking for each write - the barrier for a write must be kept until all previous writes

Response Generation to Initiator Agent

- All responses within an AXI read or a write flow (same ARID or AWID), must be generated in the order the transactions arrive at the NCAIU. This property must be guaranteed, independent of their completion order at downstream agents.
- The earliest time a transaction may be responded to is a function of the policy programmed into GPRAR [4:3], the arrival of messages (StrReq, DtrReq, DtwRsp) from downstream agents and the type of transaction (read/write).
- Responses may be generated when the event message arrives and no older transaction with the same AxID is outstanding (issued and waiting to complete, or completed and waiting to generate a response)

Type	Event	Response	Description
Read	DtrReq	RRESP	Endpoint ordered, responses may be generated when the event message arrives and no older transaction with the same AxID is outstanding (issued or completed and waiting to generate a response)
Write	DtwRsp	BRESP	

Table 19 Device Transactions (AxCACHE[3:0] = 4b000x)

Type	Event	GPRAR[4:3]	Response	Description
Read	DtrRsp	3b10	RRESP	Transactions are issued and completed in order at the downstream agent, data will be returned and saved in a reorder buffer
Read	DtrRsp	3b11	RRESP	
Write	DtwReq	3b10	BRESP	Ordered response, may be sent as soon as the responsibility for the transaction, including data, has been transferred to the completer
Write	DtwRsp	3b11	BRESP	Endpoint ordering, response will only be sent after the transaction has completed at the endpoint

Table 20 Memory Transactions (AxCACHE[3:0] = 4bxx1x)

AXI spec "Ordered write observation" says it "can support the Producer/Consumer ordering model with improved performance". AXI4 spec does not mention PCIE spec, but we know that the PCIE spec uses a Producer/Consumer ordering model. For instance, in a system like this:

CPU <-> PCIE Controller <-> PCIE AXI Bridge <-> AXI with Device and DDR slaves

(Device IP is connected to AXI slave data port0 and APB register port, DDR memory module is connected to AXI slave data port)

The CPU performs the follwing two operations,

- CPU writes data to DDR
- CPU writes Device APB register to start Device activity

Because PCIE memory writes (both prefetchable and non-prefetchable) are posted, i.e. without responses, PCIE AXI Bridge will perform the above two operations successively with the same ID but without waiting for BRESP. Before the data reaches DDR, Device may have seen the APB register write and starts reading the data, so the data may be old and invalid.

If "**Ordered write observation**" is supported, there will be no such a problem, because it requires the interface "**if two write transactions, with the same ID, are observed by all other agents in the system in the same order that the transactions are issued**", i.e. if APB register write is observed by Device, it will guarantee data write to DDR has been observable to Device.

5 Error Architecture

This section goes over Error handling in Ncore. This spec is derived from current implementation of Ncore 3.0 which was partially derived from Ncore 2.x. Effort has been made to reduce the number of changes from current implementation. In a later version of Ncore more changes and improvements will be introduced. The document first describes different registers and encodings and then goes into details of how these errors are handled

5.1 Error registers

Refer to CSR CPRs in in the hw-ncr repository under cpr/csr for register addresses.

Correctable Error Registers

5.1.1.1 Correctable Error Control Register (xCECR)

Register Description: This register controls the detection and interrupt signaling of Correctable Errors in the unit.

Register Fields:

Bits	Name	Access	Reset	Description
0	ErrDetEn	RW	0b0	Correctable Error Detection Enable. When set to 1, this bit enables detection and logging of correctable errors.
1	ErrIntEn	RW	0b0	Correctable Error Interrupt Enable. When set to 1, this bit enables the assertion of Correctable Error Interrupt signal when a new Correctable Error is detected and ErrCount value equals to ErrThreshold.
3:2	Reserved	-	-	-
11:4	ErrThreshold	RW	0x00	Correctable Error Threshold. Number of corrected errors for which logging, and interrupt (if ErrIntEn is set to 1) is suppressed.
31:12	Reserved	-	-	-

5.1.1.2 Correctable Error Status Register (xCESR)

Register Description: This register logs information of Correctable Errors in the unit. If a new Correctable Error attempts to log this register the same cycle xCESAR is configured, this register will contain information from xCESAR.

Register Fields:

Bits	Name	Access	Reset	Description
0	ErrVld	W1C	0b0	Correctable Error Valid. This bit is set when a Correctable Error is detected and ErrCount equals to ErrThreshold. xCESR, xCELR0 and xCELR1 contain logged information about this Correctable Error. Write 1 to the field resets it to 0b0.
1	ErrCountOverflow	RO	0b0	Correctable Error Count Overflow. This bit is set to 1 when ErrVld is asserted, and a new Correctable Error is detected. Once the field is set, it will keep asserted until being reset. The field is reset to 0b0 when ErrVld is reset.
9:2	ErrCount	RO	0x00	Correctable Error Count. This field increments when a Correctable Error is detected. Once the field reaches ErrThreshold, the value will be frozen from that time until being reset. The field is reset to 0x00 when ErrVld is reset. If write 1 to ErrVld and a new Correctable Error is detected at the same cycle, the field will not increment for the error.
11:10	Reserved	-	-	-

15:12	ErrType	RO	0x00	Correctable Error Type. When ErrVld is set to 1, the field indicates the type of error that has been detected and logged.
31:16	ErrInfo	RO	0x0000	Correctable Error Information. When ErrVld is set to 1, the field contains additional unit-specific and error-type-specific information about the error. For example, information that could help identify the site of the error.

5.1.1.3 Correctable Error Location Register (xCELR0)

Register Description: This register logs location information of Correctable Errors in the unit. The information is unit-specific (see Error Type and Information Summary). The fields are made RW accessible to utilize this register as the alias register for error information injection purpose.

Register Fields:

Bits	Name	Access	Reset	Description
19:0	ErrEntry	RW	0x00000	Error Entry. When ErrVld is set to 1, the field contains location information of the error. If the error is from a memory array, it indicates the Entry or Set (if from a cache) number where the error occurs. Otherwise, it is the [19:0] bits of the error address.
25:20	ErrWay	RW	0x00	Error Way. When ErrVld is set to 1, the field contains location information of the error. If the error is from a cache, it indicates the Way number where the error occurs. Otherwise, it is the [25:20] bits of the error address.
31:26	ErrWord	RW	0x00	Error Word. When ErrVld is set to 1, the field contains location information of the error. If the error is from a cache, it indicates the beat number where the error occurs. Otherwise, it is the [31:26] bis of the error address.

5.1.1.4 Correctable Error Location Register (xCELR1)

Register Description: This register logs extra address information that cannot fit in xCELR0. The fields are made RW accessible to utilize this register as the alias register for error information injection purpose.

Register Fields:

Bits	Name	Access	Reset	Description
19:0	ErrAddr	RW	0x000	Error Address Higher Bits. Contains higher bit of error address if xELR0 cannot fit the whole error address.
31:20	Reserved	-	-	-

5.1.1.5 Correctable Error Status Alias Register (xCESAR)

Register Description: This register is used to inject error status into xCESR. If a new Correctable Error attempts to log the same cycle this register is configured, the information in this register will be logged in the xCESR.

Register Fields:

Bits	Name	Access	Reset	Description
0	ErrVld	RW	0b0	Alias bit for setting ErrVld in xCESR.
1	ErrCountOverflow	RW	0b0	Alias bit for setting ErrCountOverflow in xCESR.

9:2	ErrCount	RW	0x00	Alias field for setting ErrCount in xCESR.
11:10	Reserved	-	-	-
15:12	ErrType	RW	0x00	Alias field for setting ErrType in xCESR.
31:16	ErrInfo	RW	0x0000	Alias field for setting ErrInfo in xCESR.

5.1.1.6 Correctable Resiliency Threshold Register (xCRTR)

Register Description: This register contains the resiliency correctable error threshold as an indication to the functional safety controller (FSC). Exists only if the unit enables resiliency.

Register Fields:

Bits	Name	Access	Reset	Description
7:0	Restreeted	RW	0x01	Resiliency Correctable Error Threshold. See FSC documents for more information.
31:8	Reserved	-	-	-

Uncorrectable Error Registers

5.1.1.7 Uncorrectable Error Interrupt Register (xUEDR)

Register Description: This register enables the detection of Uncorrectable Errors.

Register Fields:

Bits	Name	Access	Reset	Description
0	ProtErrDetEn	RW	0b0	Protocol Error Detection Enable. When set to 1, this bit enables the detection and logging of Protocol type Uncorrectable Error.
1	TransErrDetEn	RW	0b0	Transport Error Detection Enable. When set to 1, this bit enables the detection and logging of Transport type Uncorrectable Error.
2	MemErrDetEn	RW	0b0	Memory Error Detection Enable. When set to 1, this bit enables the detection and logging of Memory type Uncorrectable Error. This field exists only when there are protected memory arrays in the unit.
3	DecErrDetEn	RW	0b0	Decode Error Detection Enable. When set to 1, this bit enables the detection and logging of Access type Uncorrectable Error.
4	TimeOutErrDetEn	RW	0b0	Time Out Error Detection Enable. When set to 1, this bit enables the detection and logging of Time Out Error.
31:5	Reserved	-	-	-

5.1.1.8 Uncorrectable Error Interrupt Register (xUEIR)

Register Description: This register enables the interrupt signaling of Uncorrectable Errors.

Register Fields:

Bits	Name	Access	Reset	Description
0	ProtErrIntEn	RW	0b0	Protocol Error Interrupt Enable. When set to 1, this bit enables the assertion of Protocol type Uncorrectable Error Interrupt signal.
1	TransErrIntEn	RW	0b0	Transport Error Interrupt Enable. When set to 1, this bit enables the assertion of Transport type Uncorrectable Error Interrupt signal.
2	MemErrIntEn	RW	0b0	Memory Error Interrupt Enable. When set to 1, this bit enables the assertion of Memory type Uncorrectable Error Interrupt signal. This field exists only when there are protected memory arrays in the unit.

3	DecErrIntEn	RW	0b0	Decode Error Interrupt Enable. When set to 1, this bit enables the assertion of Access type Uncorrectable Error Interrupt signal.
4	TimeOutErrIntEn	RW	0b0	Time Out Error Interrupt Enable. When set to 1, this bit enables the assertion of Time Out Error Interrupt signal.
31:5	Reserved	-	-	-

5.1.1.9 Uncorrectable Error Status Register (xUESR)

Register Description: This register logs information about Uncorrectable errors in the unit. If a new Uncorrectable Error attempts to log this register the same cycle xUESAR is configured, this register will contain information from xUESAR.

Register Fields:

Bits	Name	Access	Reset	Description
0	ErrVld	W1C	0b0	Uncorrectable Error Valid. This bit is set when an Uncorrectable Error is detected. xUESR, xUELR0 and xUELR1 contain logged information about this Uncorrectable Error. Write 1 to the field resets it to 0b0. If write 1 to the field and a new Uncorrectable Error is detected at the same cycle, the field is cleared.
3:1	Reserved	-	--	-
7:4	ErrType	RO	0x00	Uncorrectable Error Type. When ErrVld is set to 1, the field indicates the type of error that has been detected and logged.
15:8	Reserved	-	-	-
31:16	ErrInfo	RO	0x0000	Uncorrectable Error Information. When ErrVld is set to 1, the field contains additional unit-specific and error-type-specific information about the error. For example, information that could help identify the site of the error.

5.1.1.10 Uncorrectable Error Location Register (xUELR0)

Register Description: This register logs location information of Uncorrectable Errors in the unit. The information is unit-specific (see Error Type and Information Summary). The fields are made RW accessible to utilize this register as the alias register for error information injection purpose.

Register Fields: see Correctable Error Location Register (xCELR0)

5.1.1.11 Uncorrectable Error Location Register (xUELR1)

Register Description: This register logs extra address information that cannot fit in xUELR0. The fields are made RW accessible to utilize this register as the alias register for error information injection purpose.

Register Fields: see Correctable Error Location Register (xCELR1)

5.1.1.12 Uncorrectable Error Status Alias Register (xUESAR)

Register Description: This register is used to inject error status into xUESR. If a new Uncorrectable Error attempts to log the same cycle this register is configured, the information in this register will be logged in the xUESR.

Register Fields:

Bits	Name	Access	Reset	Description
0	ErrVld	RW	0b0	Alias bit for setting ErrVld in xUESR.
3:1	Reserved	-	-	-
7:4	ErrType	RW	0x00	Alias field for setting ErrType in xUESR.
15:8	Reserved	-	-	-

31:16	ErrInfo	RW	0x0000	Alias field for setting ErrInfo in xUESR.

Error Type and Information Summary

5.1.1.13 Correctable ErrorType and Info Table

ErrType Code	Error Type and ErrInfo Code	Error Location			
		Addr	Word	Way	Entry
0x0	Data Correctable Error • [2:0] – Storage Type ○ 3'b000 – OTT-Data ○ 3'b001 – Read Buffer ○ 3'b010 – Write Buffer ○ 3'b011 – Snoop Filter ○ 3'b100 – Trace Buffer • [7:3] – Reserved • [15:8] – Snoop Filter ID (Reserved for non-snoop filter)	MBZ*/MBZ	MBZ/N/A	MBZ/Way	Entry/Set
0x1	Cache Correctable Error • [0] – Array Type ○ 1'b0 - Tag Array ○ 1'b1 - Data Array	MBZ	Word	Way	Set
0x2 – 0xF	Reserved	N/A	N/A	N/A	N/A

*MBZ: Must Be Zero

5.1.1.14 Uncorrectable ErrorType and Info Table

ErrType Code	Error Type and ErrInfo Code	Error Location			
		Addr	Word	Way	Entry
0x0	Data Correctable Error • [2:0] – Storage Type ○ 3'b000 – OTT-Data ○ 3'b001 – Read Buffer ○ 3'b010 – Write Buffer ○ 3'b011 – Snoop Filter ○ 3'b100 – Trace Buffer • [7:3] – Reserved • [15:8] – Snoop Filter ID (Reserved for non-snoop filter)	MBZ*/MBZ	MBZ/N/A	MBZ/Way	Entry/Set
0x1	Cache Correctable Error • [0] – Array Type ○ 1'b0 - Tag Array ○ 1'b1 - Data Array	MBZ	Word	Way	Set
0x2	Native Interface Write Response Error • [1:0] - Response • [2] - Security Attribute • [3]- Eviction	Transaction Address			
0x3	Native Interface Read Response Error	Transaction Address			

	• [1:0] - Response • [2] - Security Attribute • [3]- Fill				
0x4	Native Interface Snoop Response Error • [1:0] – Response • [2] - Security Attribute	Transaction Address			
0x5-0x6	Reserved	N/A	N/A	N/A	N/A
0x7	Decode Error • [1:0] – Type ○ 2'b00: No address hit ○ 2'b01: Multiple address hit ○ 2'b10: Illegal CSR access format ○ 2'b11: Illegal DII access type • [3:2] – Reserved • [4] – Command Type (Reserved for CHI-AIU & DCE) ○ 1'b0: Read ○ 1'b1: Write • [5] – Reserved • [15:6] – Transaction ID/AXID (Reserved for DCE)	Transaction Address			
0x8	Transport Error • [0] – Type ○ 1'b0: Wrong Target Id • [5:1] – Reserved • [15:6] – Source ID (FUnit_Id)	-			
0x9	Timeout Error • [1:0] – Reserved • [2] - Security Attribute • [15:3] - Reserved	Transaction Address			
0xA	Sys Event Error • [0] – Type ○ 1'b0: Time out error on message (protocol timeout) ○ 1'b1: Time out error on interface • [15:1] – Reserved	-			
0xB	SysCo Error • [0] – Type ○ 1'b0: Time out error on message (protocol timeout) • [15:1] – Reserved	-			
0xC – 0xF	Reserved	N/A	N/A	N/A	N/A

5.2 Native Interface Error Response Logging and interrupt

Uncorrectable Error is logged when an error response is detected at the native interfaces if ProtErrDetEn = 1.

- DMI/DII logs all AXI Bresp/Rresp errors
- CHIAIU logs all CHI data/non-data response errors
- IOAIU logs all ACE CRresp errors (CResp[1] is logged into ErrInfo Response field)

An Uncorrectable interrupt is raised if ProtErrIntEn = 1.

CCMP must be completed with error CmStatus for DtwReq and DtrReq, if the first beat of data contains error else the CmStatus does not indicate an error. If following beats of data contain error, it is indicated by setting the appropriate DBad of the data beat. Native interface to CmStatus mapping is shown in Table 21.

Native interface	Native interface error	CmStatus	Description
AXI (DMI/DII)	SLVERR	0b10000011	Data error in CmStatus. As per ARM spec this error is reported when the slave cannot provide data due to various reasons
	DECERR	0b10000100	Address error in CmStatus. As per ARM Spec this error is reported when the slave cannot be found by the network
CHI (CAIU)	Data Error	0b10000011	Data error in CmStatus. As per ARM spec this error is reported when data is corrupted
	Non-Date Error	0b10000100	Address error in CmStatus. As per ARM Spec this error is reported when the response cannot be completed due to various reasons
ACE (CAIU)	On CR channel bit 1	0b10000011	Data error in CmStatus. As per ARM spec, the agent cannot complete the snoop request

TABLE 21: NATIVE INTERFACE TO CMSTATUS MAPPING

In the cases where the data is provided at the native interface the error is propagated on either DtwReq or DtrReq. When data is not provided then the error is propagated on SnpRsp or DtwRsp.

In Ncore System, data can be rotated per DW (Double Word), the beat data error information from native interface is carried in each DW, and after rotation, certain error type can overwrite the others. For example, in AXI, if the first beat sent on SMI contains double word for both SLVERR and DECERR, Address Error is injected into CMStatus.

CmStatus to native interface mapping is shown in Table 22. In the case of DtwReq and DtrReq the first beat CmStatus may not indicate error, but consecutive beats may indicate DBad in this case the consecutive beats are treated as Data Error.

Native interface	CmStatus	Native interface error	Description
AXI/ACE-Lite/Ace-Lite E/ACE (NCAIU/CAIU)	0b10000011	SLVERR	Data error is best represented by SLVERR
	0b10000100	DECERR	Address error where the slave cannot be found is best represented by DECERR
CHI (CAIU)	0b10000011	Data Error	Data error, this includes SLVERR (reported by DMI/DII) is best mapped to Data Error on CHI as Ncore does not have enough information to decode the SLVERR and it may be reported due to bad data
	0b10000100	Non-Data Error	Address error, this includes DECERR (reported by DMI/DII) is best mapped to Non-Data Error on CHI as this error is reported when the final target cannot be found.

TABLE 22 CMSTATUS TO NATIVE INTERFACE MAPPING

5.3 Transport Error Logging and interrupt

Ncore 3 units support only Wrong Target ID error. Uncorrectable Error is logged when seeing message targetID (exclude Port_ID) mismatches unit's FUnit_ID if TransErrDetEn = 1. The message is dropped. An Uncorrectable interrupt is raised if TransErrIntEn = 1.

5.4 SRAM Error Logging and interrupt

Ncore 3 supports SRAMs to be configured with following data protection capabilities.

- No protection

- Single bit parity
- Single error correction and double error detection (SECDED)

When configured with Parity or SECDED the address of the SRAM must also be protected separately.

A parity error always results in an uncorrectable error

A SEDED error may result in a correctable or uncorrectable error as follows

- Correctable error
  - Single bit data corruption
- Uncorrectable error
  - Single- or double-bit address corruption
  - Double bit address corruption

5.5 Resiliency Related Error Logging and interrupt

SMI Protection Error

- If both resiliency and SMI protection are enabled, it is FSC's responsibility to log and interrupt SMI protection error, which includes: Concerto Header Error, Concerto Message Error and Concerto Data Error.
- Upon an SMI protection Uncorrectable Error, the message is dropped by the unit and error is logged and interrupt by FSC.
- Upon a SMI Protection Correctable Error, the error is corrected by the unit and the error counter is incremented. If Error Count reaches ResThreshold and a new error is detected, the error is logged, and an interrupt is asserted by FSC.

Please see the resiliency section for FSC error handling.

In addition, following uncorrectable errors are reported to the fault checker in resiliency mode and will trigger mission fault

- Uncorrectable memory errors which include both parity and ECC
- Wrong target ID error

5.6 Address Map Decode Error Logging and interrupt

AIU and DCE detects and logs the following Uncorrectable Errors upon address decoding if DecDetErrEn = 1:

- No Address hit
  - If the target cannot be found in the address map
- Multiple Address hit
  - If multiple targets are found in the address map
- Illegal CSR access format
  - If the target is decoded as 32-bit CSR DII, and the following requirements are not met
    - 4 Byte
    - Size-aligned
    - EndPoint order
- Illegal DII access type
  - Coherent access on DII

An Uncorrectable interrupt is raised if DecErrIntEn = 1. AIU needs to complete the protocol and return error in the response.

- ACE/ACE-LITE/AXI: DECERR
- CHI: non-data error

5.7 CCMP Completion

AIU does not expect DtrReq if StrReq contains error CmStatus for a read request, and it needs to manufacture response to the processor.

When IOAIU detects an OTT data buffer uncorrectable error

- DtwReq is returned with CMStatus = 8'b10000011 (Data Error) and poison bit set. If error is not detected on first beat, then only poison bit is set (writes)
- Data that is to be stored in cache the poison bit is set
- If the data is to be sent on to the native interface i.e. R Channel, RResp is returned with SLVERR (Reads)

When IOAIU is configurated with cache:

- For a tag Uncorrectable Error during cache lookup due to native interface request, IOAIU treats the request as a cache miss and furthermore does not allocate to the cache. In this case uncorrectable error is logged in CSrs and reported via interrupt.
- For a tag Uncorrectable Error during SNPReq lookup, IOAIU issues SNPRsp with CMStatus = 8'b10000100 (Address Error).
- For a data Uncorrectable Error upon access cache data RAM, if first beat data is poisoned and DtrReq/DtwReq is returned with CmStatus = 8'b10000011 (Data Error) and poison bit set, if a data beat after the first beat is poisoned then only poison bit is set and CmStatus may not indicate error. In the case where data is to be provided on the native interface i.e. R Channel, RResp is returned with SLVERR on the appropriate data beat.

When IOAIU processes a multi-line transaction

- For read transaction each individual Cache line is for error reporting purposes is treated separately as if it was a single cache line transaction. Requirements are defined in the section above
- For write transaction the final response on the native interface must be the accumulation of all individual cache line write responses. Here Address error (DECERR) has higher precedence than (SLVERR).

When DCE gets a snoop filter look up error

- DCE issues STRReq with CMStatus = 8'b10000100 (Address Error).

When DCE gets any snoop response error

- If any snooped AIU is providing the data, the protocol is completed by the DtrReq and the error is not propagated by DCE via StrReq. (Exception for read Stash below)
- If no snooped AIU is providing the data (DtrReq) then DCE issues StrReq with propagated error CmStatus from the last received SnpRsp with error. (Exception for read Stash below)
- In the case of Read Stash where the non-target reports SnpRsp error DCE does not propagate it and issues MrdReq to complete the request
- In the case of Read Stash where the target reports SnpRsp error, DCE propogates the error on StrReq and aborts the transaction.
- In the case of Write stash and Write Unique SnpRsp errors are not propagated onto to StrReq.
- If the original snoop was an invalidating snoop type (i.e the snooped agent is required to go to invalid state) then the snoop filter is updated to invalidate that agent, for all other snoop types, snoop filter status is not updated for that agent. (In general, it is expected that the agent that responded with an error went to an invalid state, but Ncore takes a more conservative approach due to ambiguity in ARM specs and only invalidates the snoop filter for cases where the snoop type required an invalidation)

Invalidating snoop type are: SnpInvDtw, SnpInvDtr, SnpInv, SnpInvStsh, SnpUnqStsh, SnpStshUnq.

When DMI is configurated with cache:

- For a tag Uncorrectable Error during SMC lookup due to CmdReq/ MrdReq/ DtwMrgMrdReq messages, DMI issues a control propagation DtrReq message that indicates an address corruption error. In addition, the DMI manufactures appropriately sized and poisoned data payload for the Dtrreq with CmStatus = 8’b10000100 (Address Error). Since the entire data payload is poisoned, the actual content of the payload is not important. For a tag error during SMC lookup due to a prefetch message, DMI drops the message. DMI completes the CCMP Prefetch transaction by sending STRreq without an error signal.
- For a data Uncorrectable Error upon access SMC data RAM, data is poisoned and DTRReq is returned with CMStatus = 8’b10000011 (Data Error)

CMStatus error is propagated only on following responses

- SnpRsp coming into DCE, only if no other related SnpRsp issued a DTR then this is propagated on to StrReq else it is not propagated on to StrReq
- DtwRsp coming into AIUs, this is propagated on to the native interface
- MrdRsp coming into DCE, this is propagated on to StrReq

CMStatus error is propagated only on following requests

- StrReq going out of DCE
- DtrReq going out of AIU, DMI, and DII

In the case of coherent Atomic transactions, where there are two parts the coherent part (clean operation with DCE) and the non-coherent part (atomic transaction with DMI) following applies

- If an error is reported back to the initiating AIU during the first coherent part, then the error is propagated on the Native interface and the second part is not carried out. Note that the AIU is expected to issue proper response and deallocate its transaction entry
- If an error is reported back to the initiating AIU during the second non-coherent part, then the error is propagated on the Native interface. Note that the AIU is expected to issue proper response and deallocate its transaction entry

In the case of ACE and IOAIU with proxy cache configurations where WriteBacks, WriteClean, WriteEvict, Evict and cache eviction can have two parts i.e. non-coherent write to DMI and coherent update to DCE. If an error is reported back during the non-coherent transaction part, then the coherent update parts needs to be completed. In the case of ACE the error needs to be propagated on the native interface.

5.8 Data Poisoning

When seeing any DBad on DtwReq, the receiver should not log error and just issue normal CMStatus in the response. However, the data is marked as poisoned if filling into a cache and the write strobes for the whole beat are de-asserted if writing to downstream. Note that there can be cases where the CmSstaus of the DtrReq and DtwReq may not report the error but DBad may be set. This happens in cases where the error was detected after processing the first beat of data. If error is reported in the CmStatus then the whole Cache line is considered to be poisoned.

5.9 Timeout Error

Timeout counters are implemented by AIUs, DCEs and DMIs; latter two implement it specifically to handle timeout due to recall or evictions transactions respectively. Every other transaction is covered by AIUs.

Timeout Detection:

One global counter is designed to count cycles. Once the counter reaches a programmable threshold, the timeout overflow bit in all valid entries is set. This indicates the start of the period where the entry must make progress. The timeout overflow bit will be cleared if the entry deallocates. If the global counter reaches the threshold again while the timeout overflow bit is still set, then that entry has timed out. The event is logged in an error register to allow an interrupt to be raised if enabled.

Timeout Handling:

The status is logged in the CSR software will now have the option to either disable timeout and continue or treat this as a fatal error. The timeout counter will stop running, until software restarts it via CSR.

5.10 Sys event and SysCo Errors

Sys event sender can report a timeout error if it does not get a SysRsp message for the SysReq it sent out. This is also referred to as protocol timeout.

Sys event receiver can report a timeout error if it does not get an ack back on the event pins.

Sys Co handler can report a timeout error if it does not get a SysRsp message for the SysReq it sent out.

5.11 Unsupported Message Type

If a Ncore unit received an unsupported Message (for example DMI gets a snoop command), the message will not be processed and eventually trigger a timeout error.

5.12 Error Reporting in CMStatus

If multiple errors are detected in a message, NCore Unit will report control errors (eg. Address Error) over data error in CMStatus.

6 Trace and Debug

This section describes the trace and debug architecture for Ncore. Tracing can help localizing problems and errors in a NoC which may result from functional and performance anomalies. It requires taking a snapshot of traffic in a live system with temporal order to understand what is going on. A system in general will need to implement following functionality:

- Label: Identify and select transactions based on a set criterion like address, target etc.
- Order: Identify temporal ordering of transactions, this requires a time stamp.
- Capture: Take a snapshot of the labeled transaction at a feasible location.
- Aggregate and access: Accumulate all captured transactions at a central location and provide SW access.
- Analysis: Software is required to access the captured information and analyze it.

This specification is specified with restrictions to minimize Maestro software changes and not to add any new wires to the network, this is more of an add on to current Ncore system which does have some limitations.

6.1 Register offset

Debug and trace registers described in this document are located at 0x900 offset within each Ncore unit.

6.2 Detailed Description

A top-level overview of the system with trace and debug capability is shown in Figure 14. Existing data network is used to transport trace information. The Trace trigger block is present in all the AIUs. The capture block is present in all AIUs, DMIs and DIIs. DCE and DVE do not participate in Trace capture. Note that DCE does not connect to the data network. Trace information from other units can be used to post process and build DCE and DVE trace. The trace trigger and capture blocks together fill in the functionality of label, order, and capture. Trace accumulate block is centrally located in DVE; it can be accessed via CSRs. It fills in the functionality of aggregate and access, note that it also participates in ordering transactions. The Analysis functionality falls within the software domain and is not specified in this document for now.

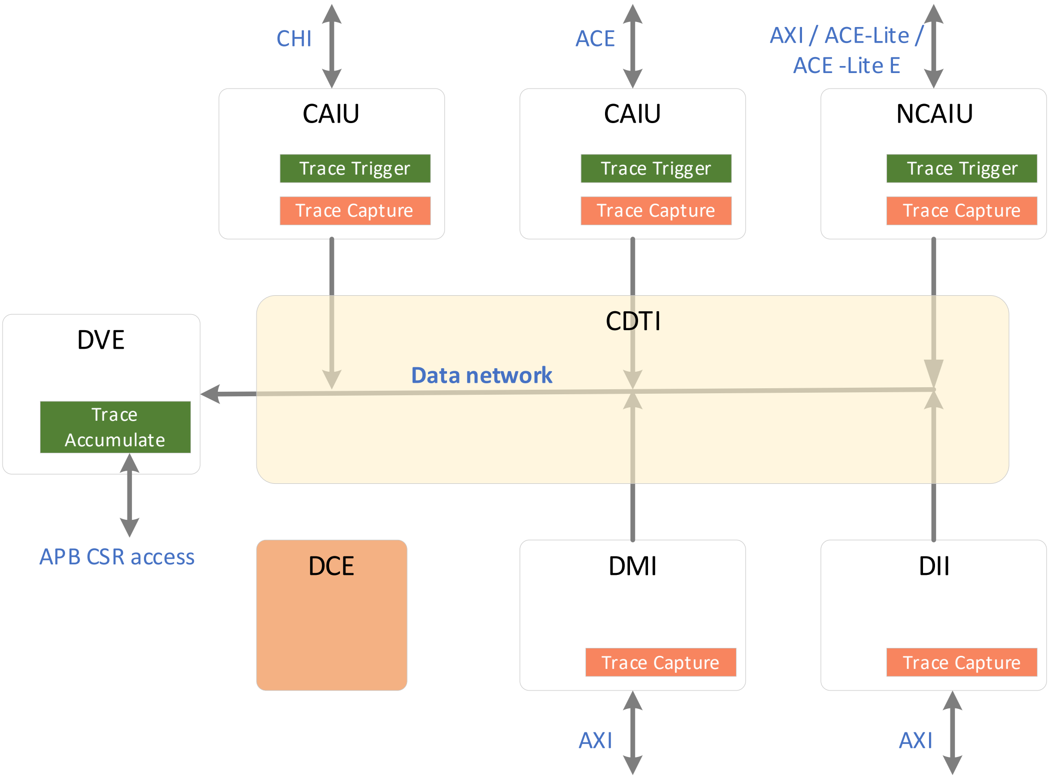


Figure 14 Trace and Debug Overview

Trace Trigger

The trace trigger block is responsible for identifying transactions to be captured and marking them. Ncore protocol has a “Trace Me” field in its messages. Ncore units shall propagate this trace signaling appropriately in all derived messages originating from a transaction that is marked to be traced (note that SysReq and SysRsp are not traced and will have their trace me field tied to zero). Furthermore, the trace signal shall be propagated on interfaces that support trace signaling based on the specific requirements of the interface (today this includes CHI-B and ACE-Lite E/ ACE5-Lite).

There are two classifications of marking a transaction to be traced.

1. Master initiated trace, this applies to CHI and ACE-Lite E (or ACE5-Lite) interfaces, which have trace signal capability on the interface.
2. Ncore initiated trace, this applies to all AIUs in Ncore, here Ncore provides CSRs which can be configured to mark desired transactions as trace transactions.

6.2.1.1 *Master Initiated Trace*

CHI and ACE-Lite E (ACE5-Lite) interfaces have trace signaling capability, the master can identify transactions that are to be traced by appropriate interface signaling. Ncroe shall honor this signaling and meet the interface requirements. In the case of ACE5-Lite the more conservative approach must be taken where all responses on the native interface must propagate the trace signal, furthermore in the case of write channel the trace signaling on W cannel is ignored and trace signaling on AW is taken into account.

6.2.1.2 Ncore Initiated Trace

All AIUs in the Ncore system including those that support trace signaling shall have the capability to initiate transaction tracing using internal CSRs. An incoming transaction on the native interface is compared with the trace CSR settings, if there is a match the transaction is marked to be traced. Multiple number of CSR sets can be present as specified at build time by the parameter nTraceRegisters. Different trace options are described below, in a single set of register all the options below can be enabled together, they are checked with an AND option i.e. that is all enabled options should match the incoming transaction. Between register sets it is an OR operation.

- **Trace Signal:** Trace transactions that are marked by the native interface, if trace signaling capability is available on the Native interface.
- **Address Range:** Trace transactions that match the specified address range. One address range can be specified per register set.
- **Op Code:** Trace transactions that match the specified Op codes. Upto 4 op codes can be specified at a time per register set.
- **Memory Attribute:** Trace transactions that match the specified memory attributes on the native interface. This is MemAttr for Chi and AxCache for AXI/ACE/ACE-Lite/ACE-Lite E. One memory attribute can be specified per register set.
- **User Bits:** Trace transactions that match the specified user bits on the native interface. One user bit value can be specified per register set.
- **Target Type:** Trace transactions that match the specified target type. The target type is specified using HUT and HUI. In the case of DII, HUI indicates DII NunitID and in the case of DMI, HUI indicates MIG number. One target type can be specified per register set.

6.2.1.3 Trace Control Register (xTCTRLR)

Bits	Name	Access	Reset	Description
0	Trace Signal	R/W	0x1	Trace using the native interface trace signaling (only present if the Native interface supports trace else it is reserved)
1	Address Range	R/W	0x0	Trace if there is a match on the address range specified
2	Op code	R/W	0x0	Trace if there is a match on the op codes specified. (Does not apply to AXI NCAIU, must be reserved in the case)
3	Memory attribute	R/W	0x0	Trace if there is a match on the memory attribute specified
4	User bits	R/W	0x0	Trace if there is a match on the user bits specified (Reserved if user bits do not exist in the configuration)
5	Target type	R/W	0x0	Trace if there is a match on the target type
6	HUT	R/W	0x0	Specifies target type DMI or DII
11:7	HUI	R/W	0x0	Specifies the HUI (NunitID for DII and MIG number of DMI)
18:12	Reserved	RO	0x0	

23:19	Size	R/W	0x0	Address range size, this field indicates a binary number from 0 to 31 from which the region's Size is calculated as (Size of IG) * 2 ^(Size +12) bytes
31:24	Memory Attribute	R/W	0x0	Specifies the memory attribute: **CHI:** bits 31:28 are MemAttr field. bits 27: 24 are reserved. **AXI/ACE/ACE-Lite/ACE-Lite E :** Bits 31:28 are AxCache field. Bits 27:26 are for AR and AW respectively, if both are set match is done on both AR and AW. Bits 25:24 are reserved

6.2.1.4 Trace Base Address Low Register (xTBALR)

Bits	Name	Access	Reset	Description
31:0	Base address LO	R/W	0x0	Lower order bits 43:12 of the base address of the region

6.2.1.5 Trace Base Address High Register (xTBAHR)

Bits	Name	Access	Reset	Description
7:0	Base address Hi	R/W	0x0	Higher order bits 51:44 of the base address of the region
31:8	Reserved			

6.2.1.6 Trace Op Code Register (xTOPCR0)

Bits	Name	Access	Reset	Description
14:0	Op code 1	R/W	0x0	Op code number 1 (mapping defined based on Native interface, MSB unused bits are RO reserved)
15	Valid 1	R /W	0x0	Valid bit for op code 1
30:16	Op code 2	R/W	0x0	Op code number 2 (mapping defined based on Native interface, MSB unused bits are RO reserved)
31	Valid 2	R /W	0x0	Valid bit for op code 2

6.2.1.7 Trace Op Code Register (xTOPCR1)

Bits	Name	Access	Reset	Description
14:0	Op code 3	R/W	0x0	Op code number 3 (mapping defined based on Native interface, MSB unused bits are RO reserved)
15	Valid 3	R /W	0x0	Valid bit for op code 3
30:16	Op code 4	R/W	0x0	Op code number 4 (mapping defined based on Native interface, MSB unused bits are RO reserved)
31	Valid 4	R /W	0x0	Valid bit for op code 4

6.2.1.8 Trace User Bits Register (xTUBR)

Bits	Name	Access	Reset	Description
31:0	User bits	R/W	0x0	Configured number of user bits rest of the field is reserved

6.2.1.9 Trace User Bits Mask Register (xTUBMR)

Bits	Name	Access	Reset	Description
31:0	User bits mask	R/W	0x0	User bits mask register, set 1 for bits that need to be used for comparison.

Trace Capture

All AIUs, DMIs and DIIs shall support trace capturing capability. The block snoops SMI interface and captures messages that have the TraceMe field set. It captures non-data request, response and data messages (this includes all NDP header fields that are of non-zero width and NDP payload); actual data within the data message is not captured. Which SMI ports are to be snooped and captured is configurable via CSR settings. Multiple SMI ports can be snooped and captured simultaneously. The capture block has a capture buffer that is sized based on the parameter nUnitTraceBufSize, this parameter specifies the number of 64-byte entries in the buffer. The captured data packed into DTWs, once enough data is captured to build a single DTW, it is sent out on the SMI data network to the DVE in the system. If enough data is not accumulated to build a single DTW then a DTW with at-least a single concerto message and padding must be sent out once a timeout is reached (the timeout value is implementation defined example 12bit counter). If the capture buffer is full then no more messages are captured until the buffer drains to accommodate at least one more DTW.

The DTW data format example is shown in Figure 15. The example is for a 128-bit bus width. The packet always starts with a 32-bit free running counter time stamp, followed by the captured concerto message. As shown multiple full concerto messages can be packed in a single a DTW. The DTW may end with padding of 0s if a full message cannot be packed into the DTW.

The capture block will give out two events. These events indicate SMI messages that were dropped and captured in a clock cycle respectively. As there can be upto 8 SMI messages that are dropped or captured, these event maybe 3 bits wide each.

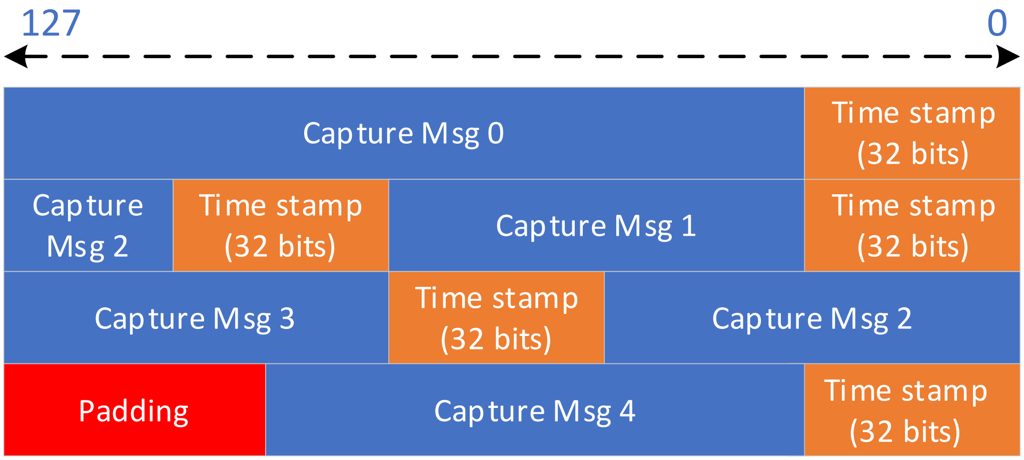


FIGURE 15 TRACE DTW DATA BUILD FORMAT

The DtwRsp cm_status field is used to synchronize the Ncore unit local time stamp counter with the DVE time stamp counter. This filed is as described in Table 23.

Field	Bits	Name	Description
cm_status	7	Signe	Positive (0) or negative (1)
	6:0	Value	Value by which the counter needs to be adjusted, this maps to counter bits [10:4] of the local counter, bits [3:0] are zero.

TABLE 23 DTWRSP CM_STATUS DESCRIPTION

The general concept is to have a feedback loop with a small correction and programable increment, as shown in Figure 16 master timeout stamp counter (MTSP) is compared with the unit timeout stamp counter (UTSC). The resultant error is multiplied with a gain and the UTSC increment value is updated accordingly.

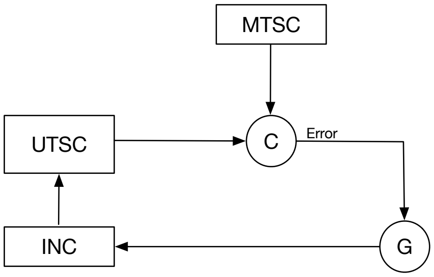


FIGURE 16 COUNTER SYNC FEEDBACK LOOP

C in the comparator of MTSC (counter in DVE) and UTSC (the local counter); comparison happens in DVE and error is sent back in DtwDbgRsp. G is the gain factor which is multiplied with the error and is used to correct INC (increment value of UTSC)

In this implementation INC is a register of 12 bits (4 bits of integer and 8 bits of fraction). This value initially represents the clock frequency difference between DVE and the Ncore unit, it gets updated based on the feedback error. Example values are shown in **Error! Reference source not found.**. The error is represented by 8 bits in singed magnitude format. where bit 8 is the sign and lower 7 bits is magnitude within DtwDbgRsp. The 7 magnitude bits are multiplied with the 4-bit gain from CSRs to give an 11-bit number. MSB 8-bits of this 11-bit number are treated as fractional value that is either added or subtracted to the INC value based on the sign of the correction.

Clock ratio with DVE	Value in INC register
2	{4'b0010, 8'b0000_0000}
1	{4'b0001, 8'b0000_0000}
0.5	{4'b0000, 8'b1000_0000}
0.25	{4'b0000, 8'b0100_0000}
0.33	{4'b0000, 8'b1100_0000}

TABLE 24 INCREMENT VALUE EXAMPLES

6.2.1.10 Capture Control Register (xCCTRLR)

Bits	Name	Access	Scope	Reset	Description
0	Ndn0 SMI TX	R/W	All	0x0	Ndn0 SMI Tx snoop and capture enable
1	Ndn0 SMI RX	R/W	All	0x0	Ndn0 SMI Rx snoop and capture enable
2	Ndn1 SMI TX	R/W	All	0x0	Ndn1 SMI Tx snoop and capture enable
3	Ndn1 SMI RX	R/W	All	0x0	Ndn1 SMI Rx snoop and capture enable
4	Ndn2 SMI TX	R/W	All	0x0	Ndn2 SMI Tx snoop and capture enable
5	Ndn2 SMI RX	R/W	All	0x0	Ndn2 SMI Rx snoop and capture enable
6	Dn0 SMI TX	R/W	All	0x0	Dn0 SMI Tx snoop and capture enable
7	Dn0 SMI RX	R/W	All	0x0	Dn0 SMI Rx snoop and capture enable
15:8	Reserved	RO	All	0x0	
19:16	gain value	R/W	All	0x2	4-bit gain value
31:20	Inc value	R/W	All	0x100	Time stamp counter increment value: top 4 bits are integer and lower 8 bits are fractional

Trace Accumulate

This block is present only in DVE and the main functionality is to accumulate incoming trace DTWs from different Ncore capture units. The capture buffer is sized based on the parameter nMainTraceBufSize. Each entry in the buffer is organized as shown in Figure 17. It is a 72-byte entry with 64-byte payload (DtwReq data received from the capture block).

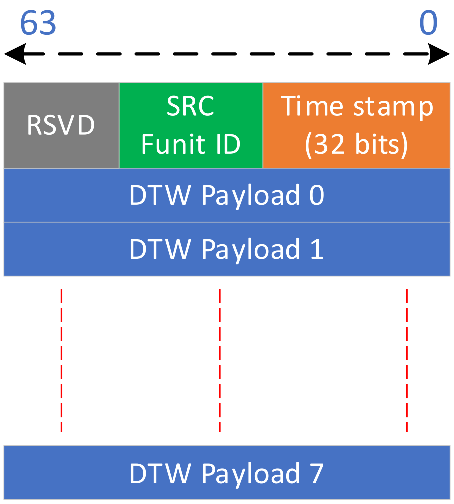


FIGURE 17 TRACE FORMAT

The time stamp here is the local free running counter at DVE. This counter is used to synchronize other Ncore unit counters. This is done by taking the first-time stamp from the received DtwReq and comparing it with bits [31:4] of the local counter, if the received time stamp value is bigger, then the cm status bit 7 of the DtwRsp is set to 1 else set to 0. Bits [10:4] of the local counter are placed into the cm status bits [6:0]. If the counter values match, then the cm status bits [6:0] are set to zeros.

The trace buffer shall be accessible via CSRs.

The accumulate block will give out an event that indicates DTW messages that are dropped per clock cycle.

6.2.1.11 Trace Accumulate Status & Control Register (xTASCR)

Bits	Name	Access	Reset	Description
0	Buffer empty	RO	0x1	Set when all the entries within the buffer are read and the buffer is empty
1	Buffer full	RO	0x0	Set when the buffer is full.
2	Circular buffer	R/W	0x0	When set, local buffer acts as a circular buffer, older messages are dropped once the buffer is full. When not set new messages are dropped when the buffer is full
3	Buffer clear	WSC	0X0	Write 1 to clear the complete buffer
4	Buffer read	WSC	0x0	Write one to trigger a read
31:5	Reserved	RO	0x0	

6.2.1.12 TTracrace Accumulate data header Register (xTADHR)

Bits	Name	Access	Reset	Description
7:0	Funit ID	RO	0x0	F unit ID of the trace capture Ncroe unit
30:8	Reserved	RO	0x0	

31	Valid	RSC	0x0	This is set once read data is valid, self clears on read

6.2.1.13 Trace Accumulate data Time stamp Register (xTADTSR)

Bits	Name	Access	Reset	Description
31:0	Time stamp	RO	0x0	Time stamp value from DVE free running counter

6.2.1.14 Trace Accumulate data (0-15) Register (xTAD0R)

These are 16 registers, where register 0 maps to data [31:0] and so on so forth.

Bits	Name	Access	Reset	Description
31:0	Capture data	RO	0x0	Capture data

7 Performance Counters

This section describes the performance counter architecture for Ncore.

A performance counter unit must be implemented in each Ncore unit i.e. AIUs, DCEs, DMIs, DIIs and DVE. This document goes over the architecture details and parameters of this unit and different events that are reported. The performance counter unit is not an optional unit and must always be present.

7.1 Detailed Description

The performance counter unit can track and report upto 31 events. The Ncore unit instantiating the performance unit specifies the events. These events maybe single bit or multi bit events and are specified in 7.3 section. The performance counter unit can be configured to implement either 4 or 8 counters.

Each counter is implemented as a 64 bit counter that is capable of counting upto 2 events at a time with following counter modes.

1. **Normal count**: In this mode the counter counts both the events together, if both the events are asserted then it is counted as two, if one event is asserted then it is counted as one
2. **AND count**: In this mode the counter counts one if both the events are asserted.
3. **XOR count**: In this mode the counter counts one only if one event is asserted and the other is de asserted.
4. **Instantaneous count:** In this mode the counter provides the instantaneous value of the multi bit event as is.

The counter reports count via 2 registers one is a 32-bit count register that reports lower 32 bits and another configurable register which supports

1. Capturing the upper 32 bits of the count
2. Used as an accumulation register for IIR filter.
3. Use as max/saturation value for accumulation events

7.2 Register Definition

Different registers supported by the performance counter unit are described below. A set of registers exist per counter configuration i.e. each Ncore unit can either have 4 or 8 set of registers

Counter Value Register (xCNTVR)

Bits	Name	Access	Reset	Description
31:0	Count value	R/W	0x0	Count value lower bits 31:0

Counter Saturation Register (xCNTSR)

Bits	Name	Access	Reset	Description
31:0	Count saturation value	R/W	0x0	Can be configured as follows: 1. Capture upper count value of 63:31 2. Use as IIR filter bits 3. Use as max/saturation value for accumulation events

Counter Control Register (CNTCR)

Bits	Name	Access	Reset	Description
0	Count Enable	R/W	0x0	Write one to enable counting
1	Count Clear	WSC	0x0	Write one to clear the counter (both CNTVR and CNTSR)
2	Interrupt Enable	R/W	0x0	Write one to enable rollover or overflow interrupt (gets ORed with correctable error interrupt)
3	Rollover/Overflow status	RO	0x0	One indicates overflow, sticky bit clears by clearing the counter
6:4	Conter control	R/W	0x0	000 – Normal count 001 – AND count 010 – XOR count 011 – Instantaneous count (used for multi bit events like Active OTT entries) 100 to 111 – reserved
9:7	SSR count	R/W	0x0	000 – Clear CNTSR (value of CNTSR is always zero) 001 – Capture upper 63:31 count in CNTSR 010 – use CNTSR as IIR filter (currently used only for active TT count) 011 – use CNTSR as max/saturation value (currently used only for CHI AIU interleave count) 100 to 111 reserved
12:10	Filter select	R/W	0x0	Low pass filter coefficients (IIR filter) 000 – 0 001 – ½ 010 – ¼ 011 – 1/8 100 – 1/16 101 – 1/32 110 – 1/64 111 – 1/128
15:13	Minimum stall period	R/W	0x0	Value is 2 ^ (minimum stall period) clock cycles valid range is 0 to 7
16:21	Count event second	R/W	0x0	Select the second count event (must be different from Count event first), value of zero is not valid
22:23	Reserved			
24:29	Count event first	R/W	0x0	Select the first count event, value of zero is not valid
31:30	Reserved			

7.3 Performance events

CAIU Performance events

Event #	Width	Name	Description
1	1	SMI 0 Tx Stall event	Counts every cycle when valid is set and ready is low
2	1	SMI 1 Tx Stall event	Counts every cycle when valid is set and ready is low
3	1	SMI 2 Tx Stall event	Counts every cycle when valid is set and ready is low
4		Reserved	
5	1	SMI 0 Rx Stall event	Counts every cycle when valid is set and ready is low
6	1	SMI 1 Rx Stall event	Counts every cycle when valid is set and ready is low
7	1	SMI 2 Rx Stall event	Counts every cycle when valid is set and ready is low
8		Reserved	
9	1	ACE AW stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
10	1	ACE W stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
11	1	ACE B stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
12	1	ACE AR stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
13	1	ACE R stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
14	1	ACE AC stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
15	1	ACE CD stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
16	1	ACE CR stall event	Counts every cycle when valid is set and ready is low (Does not apply to CHI AIU)
17		Reserved	
18		Reserved	
19		Reserved	
20	8	Active OTT entries	Number of active OTT entries
21		Reserved	
22	3	Captured SMI packets	Number of SMI packets Captured
23	3	Dropped SMI packets	Number of SMI packets dropped
24	1	Address Collisions	Count number of address collisions on incoming transactions
25	3	Interleaved Data	Count max number of active interleaved data transactions
26	1	Agent event counter	Counts number of events triggered by the native agent
27	1	Noc evet counter	Counts number of events triggered by the Noc
28		Reserved	
29		Reserved	
30	1	Div 16 counter	Divide by 16 free running counter
31	1	Number of QoS starvations	Number of times QoS starvations occurred

NCAIU Performance events

Event #	Width	Name	Description
1	1	SMI 0 Tx Stall event	Counts every cycle when valid is set and ready is low
2	1	SMI 1 Tx Stall event	Counts every cycle when valid is set and ready is low
3	1	SMI 2 Tx Stall event	Counts every cycle when valid is set and ready is low
4		Reserved	
5	1	SMI 0 Rx Stall event	Counts every cycle when valid is set and ready is low
6	1	SMI 1 Rx Stall event	Counts every cycle when valid is set and ready is low
7	1	SMI 2 Rx Stall event	Counts every cycle when valid is set and ready is low
8		Reserved	
9	1	ACE-Lite/AXI AW stall event	Counts every cycle when valid is set and ready is low
10	1	ACE-Lite/AXI W stall event	Counts every cycle when valid is set and ready is low

11	1	ACE-Lite/AXI B stall event	Counts every cycle when valid is set and ready is low
12	1	ACE-Lite/AXI AR stall event	Counts every cycle when valid is set and ready is low
13	1	ACE-Lite/AXI R stall event	Counts every cycle when valid is set and ready is low
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20	8	Active OTT entries	Number of active OTT entries
21		Reserved	
22	3	Captured SMI packets	Number of SMI packets Captured
23	3	Dropped SMI packets	Number of SMI packets dropped
24	1	Address Collisions	Count number of address collisions on incoming transactions
25		Reserved	
26	1	Agent event counter	Counts number of events triggered by the native agent
27	1	Noc evet counter	Counts number of events triggered by the Noc
28		Reserved	
29		Reserved	
30	1	Div 16 counter	Divide by 16 free running counter
31	1	Number of QoS starvations	Number of times QoS starvations occurred

Proxy Performance events

Event #	Width	Name	Description
1	1	Cache read hit	
2	1	Cache write hit	
3	1	Cache snoop hit	
4	1	Cache eviction	
5	1	Cache no ways to allocate	
6	1	Cache fill stall	
7	1	Cache read stall	
8	1	Cache write stall	
9	1	Cache replay	
10	1	Cache read miss	
11	1	Cache write miss	
12	1	Cache snoop miss	
13		Reserved	
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20		Reserved	
21		Reserved	
22		Reserved	
23		Reserved	
24		Reserved	
25		Reserved	
26		Reserved	
27		Reserved	
28		Reserved	
29		Reserved	

30		Reserved	
31		Reserved	

DMI Performance events

Event #	Width	Name	Description
1	1	SMI 0 Tx Stall event	Counts every cycle when valid is set and ready is low
2	1	SMI 1 Tx Stall event	Counts every cycle when valid is set and ready is low
3	1	SMI 2 Tx Stall event	Counts every cycle when valid is set and ready is low
4	1	SMI 3 Tx Stall event	Counts every cycle when valid is set and ready is low
5	1	SMI 0 Rx Stall event	Counts every cycle when valid is set and ready is low
6	1	SMI 1 Rx Stall event	Counts every cycle when valid is set and ready is low
7	1	SMI 2 Rx Stall event	Counts every cycle when valid is set and ready is low
8	1	SMI 3 Rx Stall event	Counts every cycle when valid is set and ready is low
9	1	AXI AW stall event	Counts every cycle when valid is set and ready is low
10	1	AXI W stall event	Counts every cycle when valid is set and ready is low
11	1	AXI B stall event	Counts every cycle when valid is set and ready is low
12	1	AXI AR stall event	Counts every cycle when valid is set and ready is low
13	1	AXI R stall event	Counts every cycle when valid is set and ready is low
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20	8	Active WTT entries	Number of active WTT entries
21	8	Active RTT entries	Number of active RTT entries
22	4	Captured SMI packets	Number of SMI packets Captured
23	4	Dropped SMI packets	Number of SMI packets dropped
24	1	Address Collisions	Count number of address collisions on incoming transactions
25	1	Number of Merge events	Count number of DtwMergeMrds
26	1	Number of system visible Txn	Count number of system visible transactions
27		Reserved	
28		Reserved	
29		Reserved	
30	1	Div 16 counter	Divide by 16 free running counter
31	1	Number of QoS starvations	Number of times QoS starvations occurred

SMC Performance events

Event #	Width	Name	Description
1	1	Cache read hit	
2	1	Cache write hit	
3	1	Cache CMO hit	Operations like cleanInvlidate etc
4	1	Cache eviction	
5	1	Cache no ways to allocate	
6	1	Cache fill stall	
7	1	Cache read stall	
8	1	Cache write stall	
9	1	Cache replay	
10	1	Cache read miss	

11	1	Cache write miss	
12	1	Cache CMO miss	
13		Reserved	
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20		Reserved	
21		Reserved	
22		Reserved	
23		Reserved	
24		Reserved	
25		Reserved	
26		Reserved	
27		Reserved	
28		Reserved	
29		Reserved	
30		Reserved	
31		Reserved	

DII Performance events

Event #	Width	Name	Description
1	1	SMI 0 Tx Stall event	Counts every cycle when valid is set and ready is low
2	1	SMI 1 Tx Stall event	Counts every cycle when valid is set and ready is low
3	1	SMI 2 Tx Stall event	Counts every cycle when valid is set and ready is low
4		Reserved	
5	1	SMI 0 Rx Stall event	Counts every cycle when valid is set and ready is low
6	1	SMI 1 Rx Stall event	Counts every cycle when valid is set and ready is low
7	1	SMI 2 Rx Stall event	Counts every cycle when valid is set and ready is low
8		Reserved	
9	1	AXI AW stall event	Counts every cycle when valid is set and ready is low
10	1	AXI W stall event	Counts every cycle when valid is set and ready is low
11	1	AXI B stall event	Counts every cycle when valid is set and ready is low
12	1	AXI AR stall event	Counts every cycle when valid is set and ready is low
13	1	AXI R stall event	Counts every cycle when valid is set and ready is low
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20	8	Active WTT entries	Number of active WTT entries
21	8	Active RTT entries	Number of active RTT entries
22	3	Captured SMI packets	Number of SMI packets Captured
23	3	Dropped SMI packets	Number of SMI packets dropped
24	1	Address Collisions	Count number of address collisions on incoming transactions
25		Reserved	
26		Reserved	
27		Reserved	
28		Reserved	
29		Reserved	

30	1	Div 16 counter	Divide by 16 free running counter
31		Reserved	

DCE Performance events

Event #	Width	Name	Description
1	1	SMI 0 Tx Stall event	Counts every cycle when valid is set and ready is low
2	1	SMI 1 Tx Stall event	Counts every cycle when valid is set and ready is low
3	1	SMI 2 Tx Stall event	Counts every cycle when valid is set and ready is low
4		Reserved	
5	1	SMI 0 Rx Stall event	Counts every cycle when valid is set and ready is low
4	1	SMI 1 Rx Stall event	Counts every cycle when valid is set and ready is low
7	1	SMI 2 Rx Stall event	Counts every cycle when valid is set and ready is low
8		Reserved	
9		Reserved	
10		Reserved	
11		Reserved	
12		Reserved	
13		Reserved	
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20	8	Active ATT entries	Number of active ATT entries
21		Reserved	
22		Reserved	
23		Reserved	
24	1	Address Collisions	Count number of address collisions on incoming transactions
25	1	SF hit	Snoop filter hit (either owner or sharer) count
26	1	SF miss	Snoop filter miss (neither owner nor sharer) count
27	1	SF recall	Snoop filter recall transaction count
28	1	Snoop rsp miss	Snoop response reports miss
29	1	Snoop rsp Owner transfer	Snoop response Ownership transfer
30	1	Div 16 counter	Divide by 16 free running counter
31	1	Number of QoS Starvations	Number of times QoS starvations occurred

DVE Performance events

Event #	Width	Name	Description
1	1	SMI 0 Tx Stall event	Counts every cycle when valid is set and ready is low
2	1	SMI 1 Tx Stall event	Counts every cycle when valid is set and ready is low
3	1	SMI 2 Tx Stall event	Counts every cycle when valid is set and ready is low
4		Reserved	
5	1	SMI 0 Rx Stall event	Counts every cycle when valid is set and ready is low
4	1	SMI 1 Rx Stall event	Counts every cycle when valid is set and ready is low
7	1	SMI 2 Rx Stall event	Counts every cycle when valid is set and ready is low
8		Reserved	

9		Reserved	
10		Reserved	
11		Reserved	
12		Reserved	
13		Reserved	
14		Reserved	
15		Reserved	
16		Reserved	
17		Reserved	
18		Reserved	
19		Reserved	
20	8	Active STT entries	Number of active STT entries
21		Reserved	
22	1	Captured DtwDbgReq packets	Number of DtwDbg packets captured
23	1	Dropped DtwDbgReq packets	Number of DtwDbg packets dropped
24		Reserved	
25		Reserved	
26		Reserved	
27		Reserved	
28		Reserved	
29		Reserved	
30	1	Div 16 counter	Divide by 16 free running counter
31		Reserved	

8 Resiliency

Ncore resiliency implementation for all Ncore units is outlined here. The top-level system view is shown in Figure 18.

At system level Ncore implements following resiliency features

- Unit duplication with 1 to 4 cycle delay
- Native interface protection via place holder
- Transport protection on SMI interface, this can be parity of SECDED.

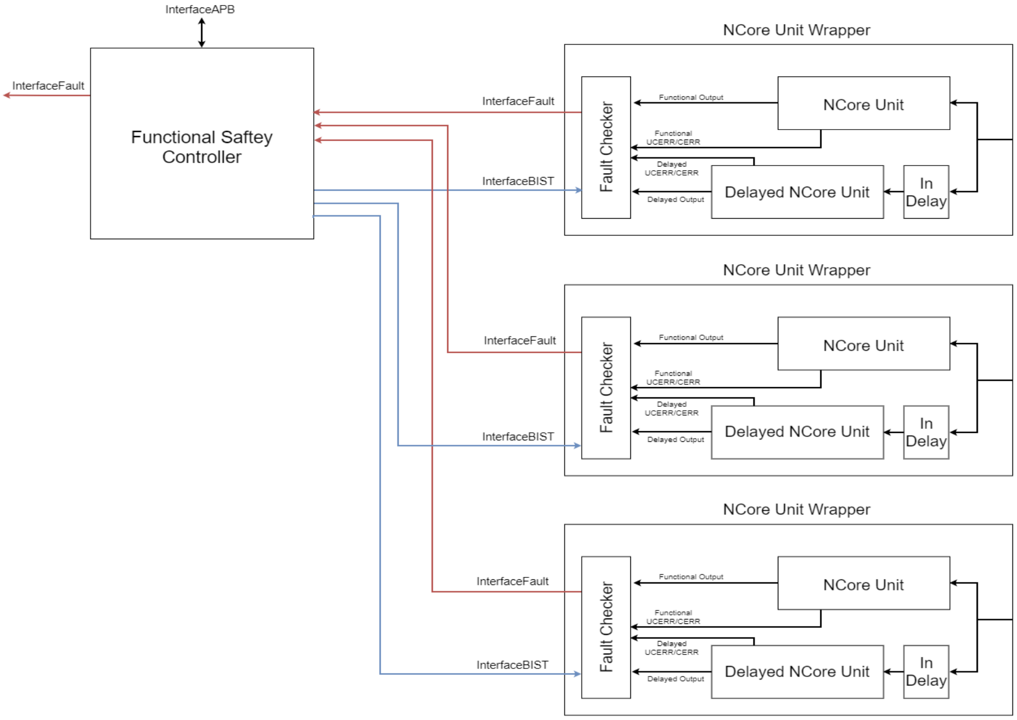


FIGURE 18 RESILIENCY TOP-LEVEL SYSTEM VIEW

8.1 Functional Safety Controller

The functional Safety Controller has two operations.

- Performs BIST Functionality over the Fault Checkers
- Accumulates the InterfaceFault interfaces into a single master InterfaceFault

See FSC Micro Architecture specification for more details

8.2 Fault Checker

The fault checker takes in four interfaces.

1. The functional outputs.
2. The delayed outputs.
3. The functional CERR/UCERR signals.
4. The delayed CERR/UCERR signals.

and then drives an InterfaceFault to the safety controller. See fault checker Micro Architecture specification for more information on how this block accumulates these signals and drives InterfaceFault.

8.3 Input Delay

An input delay unit is a simple unit which delays all inputs to the block by a parameterizable number of cycles. This is used to drive the delayed unit. See checker delay Micro Architecture for more detail.

8.4 Correctable Error Threshold

Each block will have a register controlling its correctable error threshold specifically for reporting to the FSC. (Each block may also have a register for a correctable error interrupt threshold). This value is driven to the Fault Checker. In addition, this value must be part of the interfaces that are checked by the fault checker. The fault checker uses this value to count correctable errors before it asserts a cerr_over_thresh signal to the FSC.

8.5 Memories

A Ncore unit might have external memories. If this is the case it will instantiate it inside the top wrapper and not within the unit itself. The signaling to the memories will be included in the fault checker inputs like all other signals. The signaling from the memories will be delayed and driven to the delayed unit like all other inputs.

9 Power and Clock

Ncore clock and power are defined in terms of regions, domains, and sub domains.

Regions are defined as follows:

- A power region represents a group of elements that run off a power supply that is driven by one source.
- A clock region represents a group of elements that are clocked by a single source. Clock regions are considered Asynchronous to each other.

Domains are defined as follows:

- A power domain represents a group of elements whose power can be turned on and off.
- A clock domain represents a group of elements whose clock can be turned on and off. A Clock domain may be defined at build time as synchronous or asynchronous with other clock domains within the clock region.

Sub Domains are defined as follows:

- There are no power sub domains.
- A clock sub domain is a group of elements in a clock domain whose clocks can be turned on and off dynamically as a part of logic function such as clock divider. These are not supported in the current version of Ncore i.e., it supports only one clock subdomain per clock domain, and it cannot be divided down.

The parent child relationship of regions, domains and sub domains is shown in Figure 19. A Clock region is associated to a power region. A clock domain is associated to a power domain within the same parent child relationship.

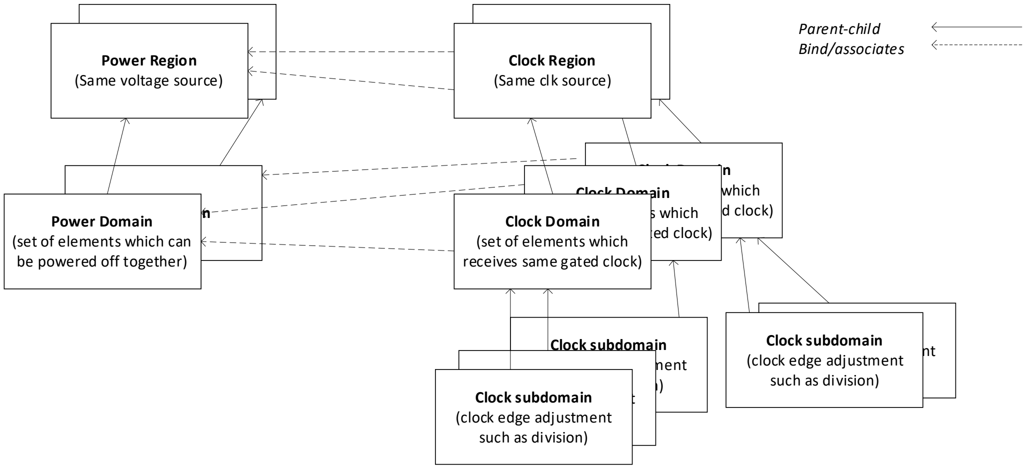

Power Region
(Same voltage source)
Clock Region
(Same clk source)
Parent-child
Bind/associates
Power Domain
(set of elements which can be powered off together)
Clock Domain
(set of elements which receives same gated clock)
Clock subdomain
(clock edge adjustment such as division)
Clock subdomain
(clock edge adjustment such as division)

FIGURE 19 POWER AND CLOCK, PARENT CHILD RELATIONSHIP

9.1 Synchronization

Synchronizing logic is inserted at every asynchronous crossing between two clock regions. As clock domains can be defined as synchronous or asynchronous, synchronizing logic is only inserted if two clock domains are defined as asynchronous.

Synchronizing logic may include any one or combination of the following:

- Multiple flopping of a single bit signal
- Request Acknowledgment handshake protocol
- Synchronizing FIFOs using gray counters

9.2 Power Gating

Ncore does not support power gating. It supports only a single Power domain. It does not provide a unified power format (UPF) file that describes level shifters and clamping cells. A user may support multiple power domains by defining UPF file by themselves.

9.3 Clock Gating

Ncore supports two levels of clock gating.

First level of clock gating is done at the flop level. These clock gates are expected to be inserted by the synthesis tool. It is recommended to gate all collection of flops which can be grouped into three or more. RTL is expected to provide enable pins on these flops to achieve this. First level of clock gating is configurable as the user may decide not to enable the option during synthesis

Second level of clock gating is at per Ncore unit basis. This is a single clock gate instantiated within each Ncore unit, It gates the whole Ncore unit except external interfaces and synchronizers. It is enabled when the Ncore unit is idle and does not have any transactions pending within itself. Second level clock gating is configurable via build time parameters as specified in the parameter spec.

9.4 Q channel

The top-level system view with Q channel support is shown in Figure 20. Note that the view is only for a single clock domain implementation. In the case of multi clock domain implementation there will be one PMA master / Q channel interface per domain.

Q channel in Ncore can be used to provide additional clock gating capability where the clock gate exists in users' logic and is not implemented by Ncore. When this capability is used, two synchronous domains must be created, one that is always on and another that can be dynamically clock gated with the help of a Q channel. This is required to make sure that the CSR network connected to Ncore is always on and does not cause CSR transactions to be lost. Q channel clock gating requires Software mechanism as described in Q channel clock gating sequence section.

Q cannel in Ncore does not support auto-wakeup, i.e., in powered down state QACTIVE will not be asserted to initiate a wakeup request by power management unit (PMU). Note that Ncore does not implement a PMU, this is expected to be user logic. Ncore only implements PMA master and PMA slave.

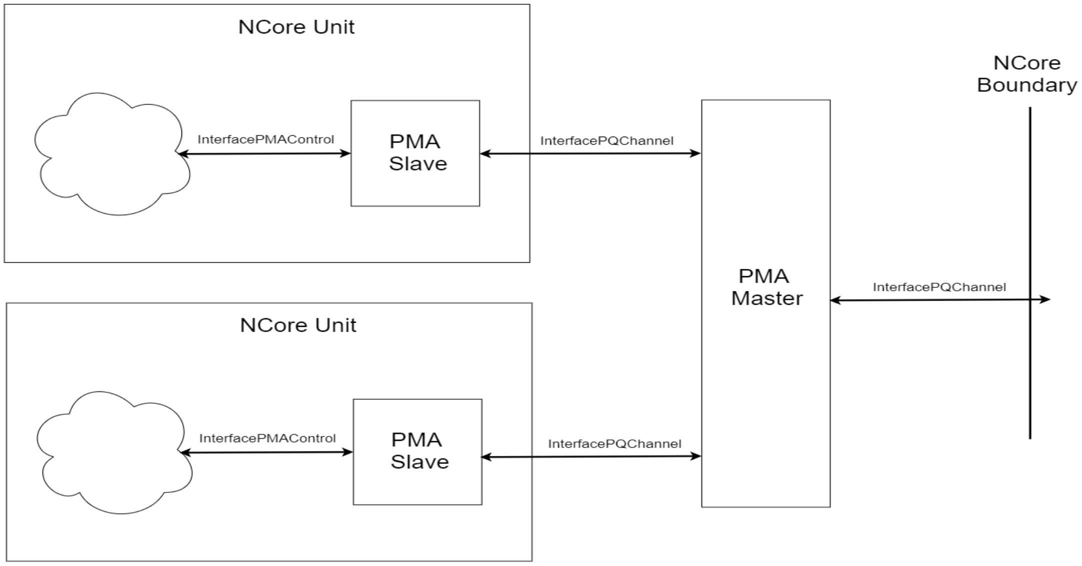


FIGURE 20 Q CHANNEL PMA TOP-LEVEL VIEW

10 System Software Guidelines

The following sections discuss the various requirements for system software.

10.1 Reset and Initialization

Before processing any requests, each Ncore component must be transitioned into the operational state, which is achieved once the following requirements have been met:

- The supply voltage has been raised to an operational level
- The clock source is generating a stable clock signal
- The clock frequency is less than or equal to the maximum frequency allowed for the given operational voltage level
- The reset signal has been asserted and has been de-asserted after the required minimum number of cycles (16 cycles of the slowest clock in the system) in the case that one or more of the following conditions occurred:
  - The supply voltage dropped below the logic or memory retention level
  - The clock source was not generating a stable clock signal
  - The clock frequency was greater than the maximum frequency allowed for a given operational voltage level

NOTE: The reset is asynchronous assert and synchronous de-assert.

Ncore components may be configured to have memories. These memories are initialized by an internal state machine on de-assertion of reset.

Once all Ncore Components are in operational state, the boot/initialization processes described below needs to be followed.

Boot/Initialization

An agent connected to CAIU/NACIU may be designated as boot agent. This agent is expected to make the first request into the Ncore system. This request is expected to be non-coherent request. Boot-up sequence to be followed are:

- Perform CSR access and go through the Ncore component discovery as described in Ncore Component Discovery
- Configure required memory and peripheral address spaces as described in Address space configuration
- Configure error registers as needed to enable required error detection and reporting as described in Error Detection and Reporting configuration
- If preset, configure and enable all needed SMCs as described in SMC configuration
- If preset, configure and enable all needed Proxy caches as described in Proxy cache configuration
- Transition CAIUs to Coherent state by the SYSCOREQ and SYSCOACK interface if present or Ncore CSRs to trigger the internal mechanism, this is described in CAIU Coherence transition mechanism.

If the Ncore system is configured with a BR then the above steps can be part of the boot code, in this case BR can be the first access to load the boot code.

Ncore Component Discovery

Ncore component discovery involves following steps at the designated CAIU/ NCAIU. All read & write operations here are at the NRS base address.

- Read register at RP `0x0` and register address offset `0x0` i.e. at address bits [19:`0`] as zero; this will read xIDR register of the AIU providing following information.
  - AIUs register page number (RPN)
  - Ncore register region identifier (NRRI), currently set to zero
  - Ncore unit ID, this uniquely identifies an Ncore unit within its type
- Read register GRBUNRRUCR at NRRI identified in the previous register, RP `0x`FF and address offset `0x`FF8. This register will identify number of different Ncore components which include AIUs, DCEs, DMIs. DIIs and DVEs. RPs to Ncore component mapping is shown in NRS section
- Step through all Ncore component xIDR registers (register address offset`0x0`) and xINFO registers (register address offset `0x`FFC) starting form RPN`0`.

Note:

1. registers access is restricted to address aligned, 8 byte non-coherent device access. Any other access format will return an error.
2. access to an undefined RPN will return an error.
3. Reserved registers within a valid RPN are treated as read as zero and write ignore.

Address space configuration

All AIUs and DCEs in the Ncore system must be configured with the same peripheral and memory address space configuration. This involves two main configuration steps as following:

- DMI interleaving configuration, this requires configuring following 2 registers
  - xAMIGR must be configured to select desired MIGS, by default MIGS`0` is selected
  - xMIFSR must be configured to select desired interleaving functioni.e., address bits used for DMI interleaving, by default option `0`is selected
- GPAS configuration, this requires configuring following 3 registers per address region
  - xGPRAR must be configured with attributes of the address region, these include
- Home unit type (HUT), this specifies type of memory. System memoryis mapped to a DMI and peripheral memory is mapped to a DII.
- Home unit identifier (HUI), this specifies target DMI or DII. In the case ofDII the Nunit ID must be specified and in the case of DMI MIG number within the selected MIGS in xAMIGR must be specified.
- Size, this is specified as a binary number from `0`to 31 from which the region size is calculated as (size of IG) * 2^(size+12) bytes. Here size of IG is the number of DMIs within the selected MIG, it is always 1 for DII.
  - xGPRBLR must be configured with lower order address bits 43:12 of the base address of the region
  - xGPRBHR must be configured with higher order address bits of the base address of the region
- ReadID/WriteID, this is applicable only to NCAIUs and specifies ordering override. When these bits are set

individually or together respective read or write channel Same AXI-ID ordering is ignored when transactions areissued into the Ncore system by the AIU. AXI-ID ordering protocol at the Native interface is honored

- Policy, this is applicable only to NCAUs and specifies the ordering policy followed by DMI/DII at the bottom of the Ncore. For details on the policies, refer to the section, PCIe Compatibility
- If present, configure address translation registers <specify register name>in DMI/DII.

Error Detection and Reporting configuration

Following steps must be followed to configure error detection and reporting for each Ncore unit/component

- Enable uncorrectable error detection by setting the appropriate error detection enable bits in xUEDR register of all Ncoreunits
- If desired enable uncorrectable error interrupts by setting theappropriate error interrupt enable bit in xUEIR register of all Ncoreunits
- Configure the correctable error control register xCECR to enablecorrectable error detection, interrupt and update the threshold asneeded
- If desired timeout error threshold can be updated or disabled atxTOCR register of all Ncore units

Proxy cache configuration

Following steps must be followed to configure and enable proxy cache

- Read register XAIUPCISR to verify both tag and data RAM are initialized
- Configure register XAIUPCTCR to enable the proxy cache in twosteps
  - Enable cache look up by settingLookupEn
  - Enable cache allocation by settingAllocEn

SMC configuration

Following steps must be followed to configure and enable SMC

- Read register DMIUSMCISR to verify both tag and data RAM areinitialized
- Configure register DMIUSMCAPR to specify desired cache allocationpolicy
- If the DMI was configured with scratchpad support, then as desired enable the scratchpad as described in Scratchpad configuration
- If the DMI was configured with way partitioning capability configure registers DMIUSMCWPCRx to setup way partitioning
- Configure register DMIUSMCTCR to enable the SMC in twosteps:
  - Enable cache look up by settingLookupEn
  - Enable cache allocation by settingAllocEn

Scratchpad configuration

Following steps must be followed to configure and enable the scratchpad

- Configure DMIUSMCSPBR0/1 with desired scratchpad base address
- Configure DMIUSMCPCR0 with desired number of ways minus 1 to be used as scratchpad. Maximum value is limited by the total number of ways configured in the cache
- Configure DMIUSMCPCR1 with scratchpad size. The size is to be specified in number of cachelines and must be an integer multiple of number of sets in the cache times the desired number of ways for scratchpad size minus 1
- Configure DMIUSMCPCR0 to enable scratchpad by setting ScPadEn

CAIU Coherence transition mechanism

There are two ways to transition a CAIU into coherent state.

CHI CAIU can transition to coherent state via the SYSCOREQ and SYSCOACK protocol signaling as specified by ARM.

CHI CAIU agent that does not have the SYSCO singling, ACE CAIU, Proxy cache NACIU and DVM capable NCAIU can transition to coherent state via the provided NCORE CSRs in the AIUs

- Read the xTAR register to confirm the current state of the AIU
- Set SysCoAttach in xTCR to start the transition of the AIU into coherent state
- Poll the xTAR register SysCoAttached field to confirm that the AIU is in the coherent state.

Transition to the detached state using CSR must follow the following steps:

- Read the xTAR register to confirm the current state of the AIU
- Set SysCoAttach to '0' in xTCR to start the transition of the AIU into detached / Non-coherent state
- Poll the xTAR register SysCoAttached field to confirm that the AIU is in the detached/Non-coherent state.

10.2 Q channel clock gating sequence

The Q channel clock gating sequence is as follows:

1. Software must go through the processes of turning off all traffic to the Ncore components that are within the clock domain and are going to be clock gated. Steps included are:
    a. Stop all coherent traffic at the native interface of the target Ncore compo- nents.
    b. Flush and disable all caches above the native interface or within the target Ncore components.
    c. Any CAIUs, Proxy cache and DMV capable NCAIUs must be transitioned to non-coherent/detached state either via SYSCOREQ/SYSCOACK interface signaling or via xTAR/XTCR registers.
    d. Stop all at the native interface of the target Ncore components.
2. Trigger the CMU (Clock gating Management Unit) to send a Q channel request which includes the following:
    a. REQn is asserted on the Q Channel Interface.
    b. Ncore component will assert ACCEPTn and accept the request once it is not busy (no active transactions are pending in the Ncore component).

The software or CMU is required to implement a timeout counter to cover cases where the Ncore component may not accept the power down request. In this case ACCEPTn will not be asserted, at timeout REQn must be de-asserted (Clock gating aborted) and that Ncore component stays in up state.

In the Clock gated state the Ncore components will not block any transactions and have no special behavior. Depending on the system design on successful completion of Q channel request the Ncore subsystem clock can be turned off.

The ACTIVE signal indicates if the block is busy and this signal can toggle at any time during normal operation.

The bring back sequence is as follows:

1. Enable the clock.
2. De-assert REQn.
3. Go through the process of transitioning any CAIUs, Proxy cache, and DVM capable NCAIUs to coherent state.
4. Start normal traffic.

Initial system power up with Ncore clock domains in state is not supported.

11 Opens

Questions/Feedback/Need to discuss:

12 Glossary

Arteris

A NoC Company

NCore3

A coherent NoC provided by Arteris with AMBA interfaces and built-in caches.

13 Notes

Notes ……