Ncore System Architecture Specification

Product release: 3.8.0

Revision: , September 24, 2025

Release Information

Version	Editor	Change	Date
0.495	BM	• Added addr map erro and SMI protection error for the GIU. • Added a section on UpdReq. • Did some clean up on Loop back. • Updates DCEUCAMR to have offset x800 because x500 was colliding with GPRAR when more than 16 were configured.	
	HL	• Updated section 2.12.2 to describe hierachical routing, which is employed to improve deadlock avoidance for the multi-die architecture after multiple group discussions with Benjamin and Federico	7/10/2025
0.494	BM	• Updated 3.5.2.2 to have number of register counter match the number of dies in the system.	07/09/2025
0.493	BM	• Update to the order of fields in the target Id in paragraph 2.12 to match paragraph 3.1.	06/27/2025
0.492	BM	• Update to section 3.1 to change the order of the fields inside the targetId and have the globalId easily accessible • Update to the credit register programming figure • Clarification of some sentences per the reviews.	06/10/2025
0.491	BM	• Adds a section 3.5.2.3 to provide an example of credit programming • Adds section 0. • Clarification in error chapter 4 • Update to figure Figure 2-2 Overview of routing responsibilities between the Protocol Layer and Routing layer to show the FUnitId also as a new add on for local routing. • Multiple improvement in response to DVM, attachment and event walkthroughs. • Add offset to interleaving register in Table 2-4 xGIUIFSR GIU Interleaving Function set register. Register offset : 0x4cc and in Table 2-3 XAGIUIGR: Active GIU Interleaving Register Offset : 0x4c8 •	05/19/2025
0.	BM	• Update to the coherent flow diagram which was very wrong • Update to section 3.1 : InitiatorId and TargetId and 3.2 per review on May 12th • Update to section 4 per review on Maty 12th.	05/13/2025
0.0.48	BM	• Minor typo in loopback legend • Minor correction • Update to LinkId assignment for SnpReq • Update to RemoteLinkInterleavingObject (2.5.4)	5/01/2025
0.47	BM	• Minor typos fix per Hao Feedback • Update to Table 11-1 GIULCSTR: GIU Loopback Control and Status Register • Update to Table 7-2 GIUUIDRGIU Identification register, Address offset : 0x0.	4/15/2025

		• Update to Table 11-3 An Example of Packet Based on A 64-byte Flit with A Starting Data of 32'h12345678, and a left shift of 4 bits. • Added a mention that CXS 32B streaming in not supported. • Ran the spelling and grammar checks.	
0.46	BM	• Update the requirement and specification (section 1.2.1) per Hao feedback. • Revert style of the bibliography to IEEE • List of local agents in DVE for attachment (sections 3.10.4 and 3.10.5) • Minor update to the FUSA section to account for discussion with Stefano • Add a section for GPRAR in the GIU (section 2.7.3) • **Added section 3.2 for TargetId update in the GIU. Very important architecture hole**.	4/06/2025
0.45	BM	• Added section 8 Example of partially populated chip use. • Modified the boot sequence to reuse the description of the user manual and added the link bring up as soon as possible. • Added the GIU interleaving registers (Section 2.5.3). • Minor cleaning in the loopback section (Section 13)	3/31/2025
0.44	BM	• Reshuffled the DVM/Attachment/Event section of version 0.43 in multiple separate paragraphs (3.7 ,3.9,3.10) • Updated the DVM description to account for the decision made regarding the centralized/distributed approach. (3.9) • Cleaned up the DVM related registers. • Rewrote the exclusive monitor section (3.8) • Adds clarification on data width of the GIU (2.4.3) • Adds clarification on network configuration allowed for C2c systems (2.4.2) • Moved the error reporting, fusa and trace capture section higher in the numbering hierarchy. • Update to the FUSA paragraph. (Chapter 5) • Update to the error reporting paragraph. (Chapter 4) • Update to the trace capture paragraph. (Chapter 6) • Update to the requirements to have the latest known restrictions (Chapter 1.2) • Update to the System overview to remove sentences that were related to some earlier version of the spec only (Chapter 2) • Tried to simplify the explanation of the boot sequence. (Chapter 7)	3/26/2025
0.43	HL	• Updated the boot-up flow to ensure NO individual AIU starts SysCo attachment before all physical links are up running per defined in **Error! Reference source not found.**. • Removed previous coherent, DVM and event node attachment and de-attachment diagrams since the attachment is implicit and it is carried out automatically.	3/19/2025

		• Updated event node handling as defined in **Error! Reference source not found.**. • Updated the logical coherent and DVM domain control and status register definition and usage in **Error! Reference source not found.** and **Error! Reference source not found.**. • Added **Error! Reference source not found.** to explain how different configurations can be controlled. • Clarified exclusive monitors' indexing in section **Error! Reference source not found.**. • Added **Error! Reference source not found.** based on the latest Synopsys Databook to clarify ARM CHI-C2C compliant implies from Synopsys perspective, which an Ncore has to follow. • Made GPRAR definitions consistent across IOAIU, CAIU and DCE as described in section 2.7.2. • Started to explicitly assign new-added CSRs to their corresponding Ncore units as described in section **Error! Reference source not found.**, section **Error! Reference source not found.**, section **Error! Reference source not found.** and section **Error! Reference source not found.**. • Removed current chiplet ID subfield acroases multiple CSR registers and consolidated in one register, which is located in GRB as defined in **Error! Reference source not found.**. • Went through the review feedback from Boon and Benjamin and further clarified RW/RO and RO usage for ALL new CSR registers across the specification.	
0.42	BM	• Adds 3.4 LinkId calculation to explain how it is derived for Snp, Cmd and Upd. • Adds 2.12.1.2 LinkId overflow to explain what happens to message that contains a LinkId that may not exist to access the next chiplet • Fixes minor typographic issues. • Review of 0.41 Modifications	3/7/2025
0.41	HL	Updated the following based on the deep-dive discussion with Benjamin Madon and Boon Chuan held in Campbell on March 4, 2025: • Further aligned and consolidated the Snoop Filter Assignment described in the previous section 3.3 and merged into coherent node handling in section **Error! Reference source not found.**. • Redefined snoop enable register usage as described in section **Error! Reference source not found.** • Redefined snoop enable mapping registers as shown in **Error! Reference source not found.** • Added stash enable registers as described in section **Error! Reference source not found.**. • Removed the subfield MaskBits in SysReq.ndp as shown in **Error! Reference source not found.**. And updated **Error! Reference source not found.**, **Error! Reference source not found.** and **Error! Reference source not found.** accordingly to reflect the change.	3/6/2025

		• Added section 2.7.2 to define how DCE define the LinkId for snoop request. • Moved the relevant registers used in both coherent and DVM nodes (previous section 3.5.2) ahead of their corresponding attachment/de-attachment (previous section 3.5.1) to make it more readable	
0.39	HL	• Added Table 2-7 GPRAR Definitions in Ncore 3.8 with Updates and Table 2-8 GPRAR Definitions in Ncore 3.7 for Reference to highlight the proposed changes on GPRAR • Clarified HUT reference and HUI interpretation in section 2.7.1.2. And assigned GPRAR[9:8] to maintain backward compatibility. • Introduced LinkId register for coherent, DVM and event node related traffic and added **Error! Reference source not found.**, **Error! Reference source not found.** and **Error! Reference source not found.** • Removed previous sections 2.8.2, 2.8.3 and 2.8.4 since the coherent, DVM and event agents are described in detail in the following sections. • Added section **Error! Reference source not found.** to define the event node and also updated **Error! Reference source not found.** to describe where resides. • Added coherent node registration handling per Boon's feedback to avoid potential race conditions with an individual AIU attachment in section **Error! Reference source not found.** ○ Also corrected multiple typos and misc. errors across multiple flow diagrams in section 3.9, **Error! Reference source not found.** and 3.12	3/3/2025

0.38	HL	• Added ndp.RequestGlobalID subfield per Boon's feedback in **Error! Reference source not found.**
		  ○ Updated **Error! Reference source not found.**, **Error! Reference source not found.** and **Error! Reference source not found.** accordingly.
		• Added **Error! Reference source not found.** to illustrate the combined coherent and DVM attachment flow.
		• Updated system event handling in section **Error! Reference source not found.** and added relevant registers.
		• Moved boot-up and initialization related descriptions and registers to section **Error! Reference source not found.**
		• Updated Figure 7-3 based on the most recent IP release from Synopsys.
		• Updated section **Error! Reference source not found.** to clarify and define DVM snoop enable register usage.
		  ○ Updated accordingly on DVM cross chiplet handling for **Error! Reference source not found.**
		• Added section **Error! Reference source not found.** to describe how snoop enable registers are being used and how snoop enable mapping registers are defined.
			2/25/2025

		• Further clarified the current chiplet-to-chiplet solution is a die-to-die NOT a chip-to-chiplet architecture in section 1 • Cleaned up Figure 1-1, Figure 1-2, Figure 7-1 and Figure 2-1 to reflect it is a die-to-die architecture • Corrected some minor issues in Figure 11-1	
0.37	HL	• Added Figure 2-13 UCIe Format 6 Packet Layout For Streaming to define what packet layout the current version supports and updated the limitations on CXL supports in section 2.11.4.3 • Added **Error! Reference source not found.** and updated section **Error! Reference source not found.** to define the event flow for the multi-die scenario. • Added Table 7-5, Table 7-6, and Table 7-7 to further clarified the usage of Link Connections Register defined in **Error! Reference source not found.** • Removed extra Opcodes reserved for coherent node attach/de-attachment and reuse the existing Opcodes for such purposes. • Kept the same OpCodes (0x1/2) for the SyscoReq/AcK but added DVM node attachment/de-attachment subfield in ndp as described in section **Error! Reference source not found.**. all relevant flow diagrams are updated to reflect the change. • Added to define new ndp subfields for coherent and DVM attachment/de-attachment messages. • Added a register to keep a record of the FunitIDs of all DVEs in an assembly after an assembly is fully configured as shown in **Error! Reference source not found.** • Fixed various error reference issues in section 4 due to frequent updates prior to this version. And cleaned up the corresponding descriptions to reflect coherency and DVM domain related changes	2/10/2025
0.36	HL	• Updated DVM flow control in section **Error! Reference source not found.**	12/31/2024
	BM	• Removes the concerto parameters from this document and refers to the Ncore 3.8 parameter specification.	12/23/2024
	HL	• Updated DVM transaction handlings per Boon Chun's suggestion reflected in **Error! Reference source not found.** and **Error! Reference source not found.**, which significantly reduces the number of messages across chiplets. • Updated Coherent and DVM node attachment and de-attachment flow reflected in **Error! Reference source not found.**	12/20/2024
0.35	HL	• Moved relevant coherent and DVM boot-up/attachment registers to be co-located with the coherent and DVM descriptions in section 3.9 • Added more flow diagrams on how coherent and DVM nodes work in attachment and in normal operation. • Added the details of the DVM node functionalities in section 3.9 • Cleaned up and sections across the specification and removed PCIe support and relevant descriptions.	12/18/2024

		• Further clarified registers' access in section **Error! Reference source not found.** should be privileged. • Corrected a few typos based on Benjamin's review feedback	
0.33	BM	• Add an overview in section 2.3 to give high level description of the routing mechanism with reference to relevant paragraphs for the details.	11/22/2024
	HL	Further cleaned up the boot-up and initialization process with more detailed and confirmed information from Synopsys on UCIe controller: • Added a zoom in diagram as Figure 7-1 to show the necessary SoC level collateral components to fulfill and complete the initialization flow. • Added Figure 7-2 to detail necessary interfaces among multiple dies within an assembly to complete the initialization flow. • Cleaned up Figure 7-3 based on the latest information from Synopsys. • Added Figure 7-5 and Figure 7-6 to show the initialization flow for an assembly with four chiplets • Added two registers inside the coherent node and DVM node to control and monitor the coherency and DVM domain registrations	11/20/2024

	BM	Per the workshop feedback and further discussion:
		• Update to the GIU grouping section to account for routing inside Legato (2.5)
		• Update to the routing layer to locate back the global routing function inside the routing layer description (2.12)
		• Update to the protocol section (0)
		  ○ Removes the message update which became unnecessary.
		  ○ Update to the SnpReq credit to allow for global credit. Describes a proposed mechanism.
		  ○ Update to the CmdReq credit to allow for global credit. Describes the proposed mechanism.
		  ○ Adds a description for InitiatorId and TargetId fields.
		  ○ Minor update to the Snoop filter assignment (not consequential)
		  ○ Update to the ordering.
		  ○ Update to the VC mapping to use SMI
			11/7/2024

0.322	BM	• Update to the packing algorithm to account for the choice made during the workshop. • Added message structure and granule number for each Ncore VC	11/01/2024
0.321	BM	• Update to the address map to describe the revised mechanism. • Rework the interleaving section to describe the revised mechanism. Added a grouping section and described the concept of Route Group (RG) and LinkId.	10/24/2024

		• Modified the InterleavingId into a LinkId (diagram not updated yet)	
	HL	• Added ARM GIC-700 support in section 10	10/25/2024
	HL	• Added a flow chart for initialization flow for UCIe based solution	10/29/2024
0.32	HL	• Added coherent and DVM domain attach/de-attachment flows. • Added DVM message flow diagrams. • Updated initialization flow	10/23/204
0.31	BM	• Writing the protocol Control section • Added more description in the Address map section	10/18/2024
0.30	BM	• First clean-up version after re-mapping routing changed to purely global routing. Ported chapter 1 and part of chapter 2 from previous document. • Reworked some of the address map description. • Reworked the interleaving description	10/04/2024
0.1	SD	• Initial version	
	SD BM HL Xx	Said Derradji Benjamin Madon Hao Luan Whoever else edited this document	

Confidential Proprietary Notice

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 intends 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.

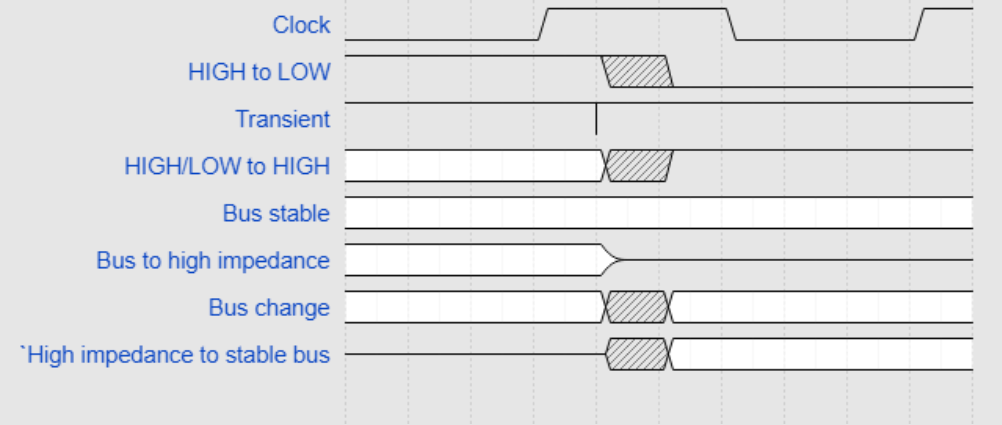
SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, which 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.

1 Introduction

1.1 Preface

The scope of this specification is a chiplet-to-chiplet/die-to-die extension of the Ncore 3 system architecture specification.

The scope of this document is to specify a chiplet-to-chiplet or a die-to-die NOT a chip-to-chip architecture even though some key aspects of the extension to chip-to-chip have been considered in this specification.

Every agent in the system has the visibility of all remote agents participating in the cache coherency messaging protocol. The remote agents are defined as agents sitting in a different chip(let) behind a die-to-die link. Ncore proprietary cache coherency messaging protocol is tunneled across the die-to-die link. An Ncore instance, GIU (Gateway Interface Unit), is expected at each side of the link, and the same 3rd party PHY and corresponding controller are also expected to be used across all chiplets.

No discovery mechanism is expected to exchange across links. The visibility shall be statically set at boot time.

1.2 Supported configurations

The targeted configurations range from symmetric/homogenous configuration as shown in Figure 1-1 to asymmetric/heterogenous configuration as shown in Figure 1-2. This means that every Ncore instance requires at least a directory, a memory interface, and a coherent agent.

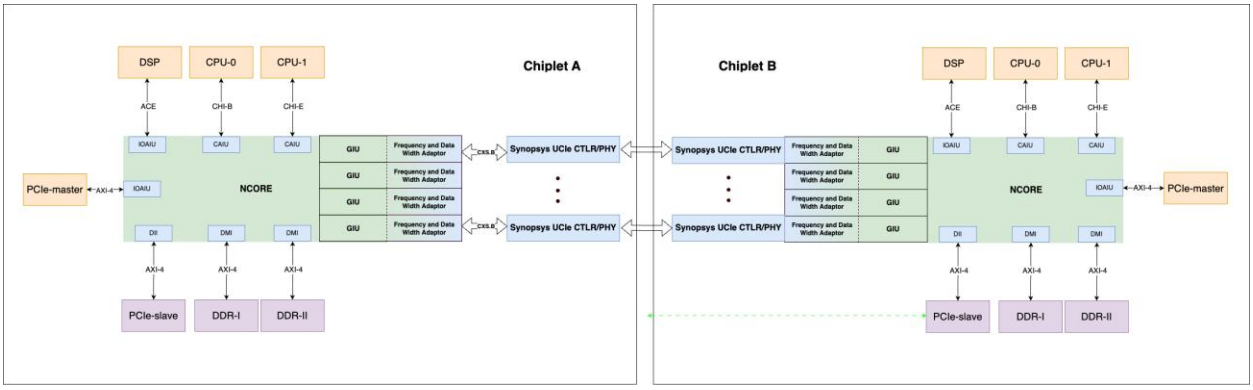


FIGURE 1-1 SYMMETRIC CHIPLET USAGE OF AN ASSEMBLY

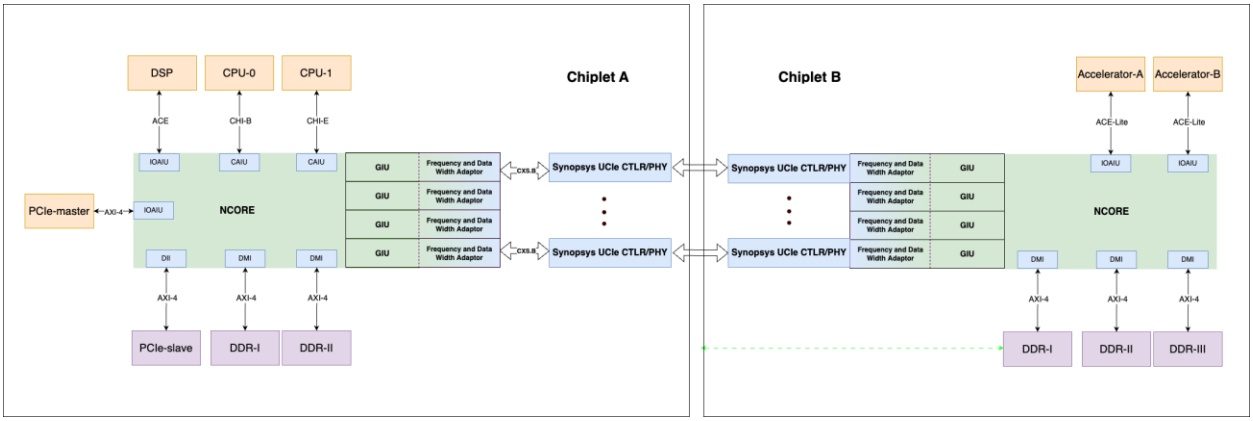
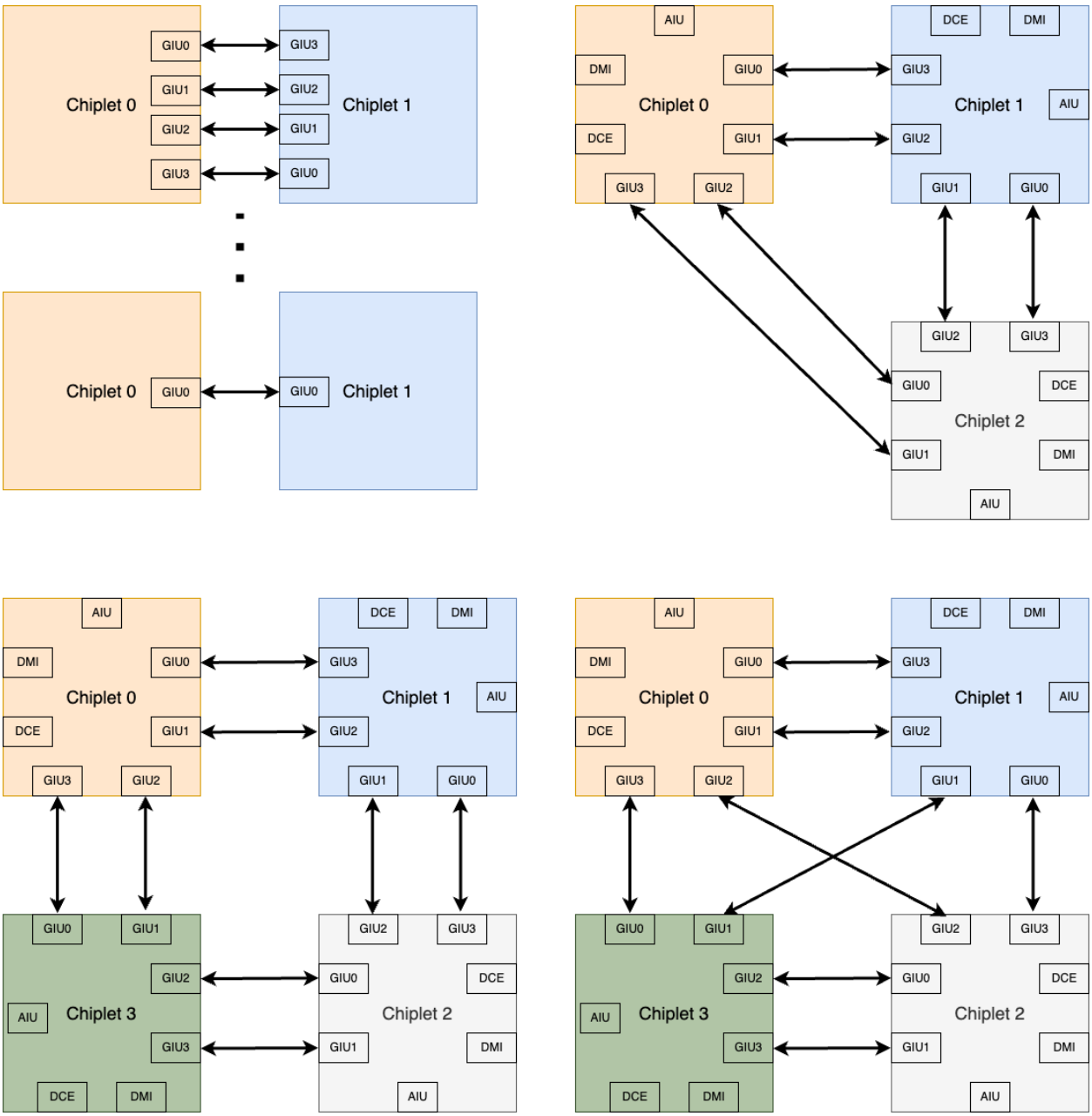


FIGURE 1-2 ASYMMETRIC CHIPLET USAGE OF AN ASSEMBLY

In addition, multiple SOC variants may be built using several instances of the same or different chiplet(s). Figure 1-3 shows the supported topologies and their corresponding connections. There will be some compatibility rules described further in the document. Such assembly will need to be checked by Maestro at design time and there is no guarantee that the unchecked assembly will work.



33 FIGURE 1-3 SUPPORTED TOPOLOGIES AND THEIR CORRESPONDING CONNECTIONS IN AN ASSEMBLY

1.2.1 Requirement and limitations of Ncore 3.8.0 release

In the requirements below an assembly is a collection of dies connected through UCIe links.

- **The Ncore 3.8.0 D2D uses CXS.B** to connect to the Synopsys UCIe/CXS controller.
- **Synopsys** UCIe/CXS.B controller in CXS 256B flit streaming / 64B interfaces is the **only** one supported for which compatibility will be guaranteed.
- **No reuse** with later designs
- Only **one** chip/Assembly per design.
- **Limited support** for configurations in which a **subset** of the assembly is used.
- All Dies/Chiplet must be able to **boot independently.**
- Maximum **4 CXS.B ports** per Ncore, configured per the Synopsys controller.
    - Each port connects to one GIU.
    - Each GIU connects to one SMI port per network.
- **No interoperability** is possible with dies not using Ncore 3.8.0.
- **No back compatibility or forward compatibility**: Ncore 3.8.0 are required on both sides of the D2D link.
    - All chiplets must be co-designed and co-verified within the same project and at the same time.
- **No discovery or parameters negotiation** is supported. A valid system configuration shall be set at boot time on both sides.
- **The SMI messages shall be tunneled**. The messages exchanged between all Ncore 3.8.0 instances in an assembly are private. The content of the transactions exchanged over the tunnel is not exposed outside of the Ncore system.
- The D2D links partition a global system into several dies**. It is required that every sub-system be a valid Ncore 3 standalone system. It means that:**
    - **Every Ncore 3.8.0 die must have a DVE.**
    - **Every Ncore 3.8.0 die must have a SYSDII.**
    - **Every Ncore 3.8.0 must have a GRB.**
- **Scalability**: Up to **4** chiplets. Each Chiplet has the same restriction as a standalone Ncore 3.7 system. Note that there **may be bottlenecks not addressed** in this specification (e.g., ATT size, OTT size, overall, SMI bandwidth etc.…) which may limit the practical max system.
- DCE home and DMI shall be in the same chiplet for each address space.
- Maximum of **4 D2D links** per chip(let). There is a 1 to 1 correspondence between links and GIUs.
- Chiplet topologies can be mesh or fully connected.
- No support for two or more Ncore connected without the Synopsys controller in between.
- The same chiplet can be replicated multiple times.

1.2.2 Features considered for subsequent releases

- Support for any CXS.B compliant controller
- Multiple assemblies
- Reuse of chiplets
- Partial Ncores
- Split CmdReq credits
- Split Snoop credits
- Error handling from remote chipelts

2 System overview

2.1 Gateway Interface Unit (GIU)

A new D2D gateway unit GIU is added in front of the link. It is responsible for merging all physical networks into one link implementing virtual channels to maintain independent progress between the physical networks. It is also responsible for packing the SMI messages into the selected FLIT format.

Several GIU units can be instantiated to provide several D2D links but **only one physical link per GIU is supported**.

The GIU has very limited protocol responsibilities. Its main responsibilities are to:

- Interact with the 3rd party controller to make sure the die-to-die link is properly initialized.
    - Interpret and transmit necessary customized handshake flits with the 3rd party controller.
- Pack the concerto messages and generate the compatible flits based on an interface protocol such as CXS and transmit to the 3rd party controller.
- Receive flits from the 3rd party controller, de-pack the flits and convert them to concerto messages.

Please note that there must be one valid standalone Ncore on each chiplet.

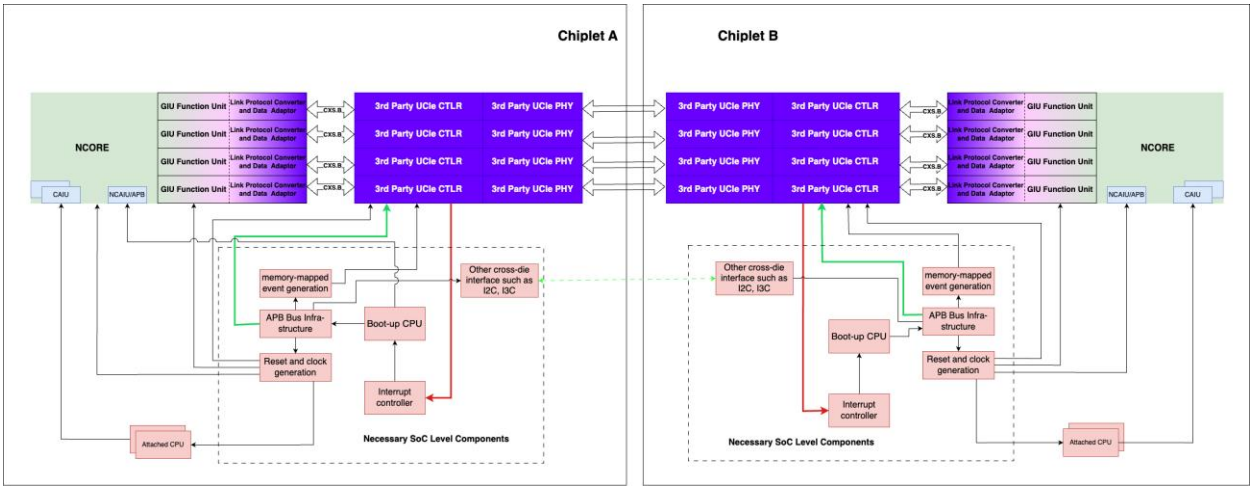


FIGURE 2-1 AN SOC INTEGRATION VIEW OF DIE-TO-DIE SOLUTION

2.2 System configurations and target performances

As shown in Figure 1-3, the following summarizes the system configuration and its target performance:

- Synopsys UCIe controller, its corresponding PHY for standard or advanced package are employed to complete the end-to-end data path.
- 16GT/s data rate is targeted.
- Each GIU link can provide up to 128 GB/s unidirectional raw throughput with 64 lanes of UCIe-A
- Each GIU interfaces with Synopsys UCIe controller and PHY via a CXS.B interface
    - With 512-bit data interface at 2 GHz (64 lanes of UCIe-A setting)
    - With 512-bit data interface at 500 MHz (16 lanes of UCIe-S setting)

2.3 High level view of routing in a multi-die context

As described in subsequent chapters, the GIU doesn't have an active role at the protocol level. Consequently, some modifications are required to the address map and to the routing layer. This paragraph is meant to serve as an introduction to those concepts and to provide references where the new concepts are described in more detail.

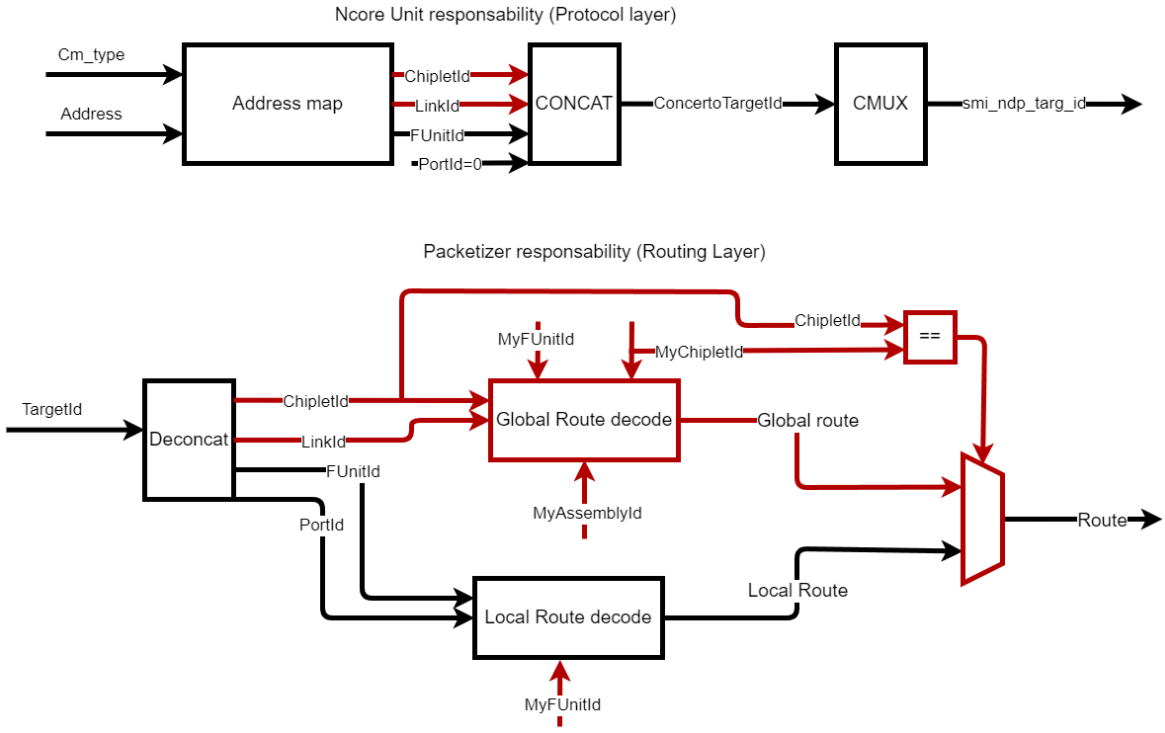


FIGURE 2-2 OVERVIEW OF ROUTING RESPONSIBILITIES BETWEEN THE PROTOCOL LAYER AND ROUTING LAYER

Figure 2-2 shows the updates that are needed to route a message to another die. The elements in red are new additions for the multiple-chiplet architecture.

At a high level, the address map decoder of an Ncore unit creates two new fields called ChipletId and LinkId which are appended to the TargetId and InitiatorId. Two new fields are introduced inside the packetizer along with ChipletId and AssemblyId tie-offs to generate a route. This route can be either local meaning the corresponding traffic does not cross a chiplet boundary or global meaning it goes to a GIU and crosses the chiplet boundary.

More details are given in section 2.5, which has the descriptions of the LinkId and how GIU are grouped and interleaved; Section 2.6 and 2.7 describe the updates made to the address map in order to generate those new fields; section 2.12 describes the packetizer with the new fields and how to obtain a route in the routing layer. Section 3.1 explains the modifications made to the TargetId and InitiatorId for the protocol layer and section 3.12.3 shows an example of the TargetId field update when the message crosses the chiplet boundaries.

Note: For Ncore 3.8.0, the AssemblyId will be tied to 0 within Ncore.

2.4 Maestro system constraints and visibility

2.4.1 Coherent agent visibility

As Ncore 3 does not support hierarchical coherency, it is required that Maestro comprehends the full list of caching agents participating to the coherency protocol. As caching agents may reside in any chiplet, Maestro shall have the global visibility of the system.

Moreover, Ncore 3 defines a mechanism to attach/detach coherency agents. The mechanism shall be extended with agents from remote chiplets. There isn't any discovery mechanism for multi-die architecture. The list of agents preexists and is initialized by Maestro.

The maximum number of coherent units shall be known at configuration/build time because it directly drives the sizing of hardware resources in the directory for instance.

Note that, **although it is not allowed by this specification to use a chiplet in an assembly it was not designed for**, another approach would be to oversize the resources provisioning for more agents that may be added when connecting remote chiplets. This approach would allow a bottom-up definition of each partition independently from the others. Then several systems may be built using Ncore 3 partitions as building blocks. **The limited resources of each partition would constrain the maximum feasible system configuration.**

2.4.2 Constraints on the number of networks

A system which supports multiple die/chiplet must use the 4CN/1DN configuration. This is decided for two reasons:

1) Use efficiently the bandwidth provided by the D2D link.
2) Simplify design and definition of the virtual channels.

2.4.3 Constraints on the width of the SMI data port going to/from the GIU

The GIU will always have an SMI data width of 512 bits. It is expected that it connects to a switch with a data width of 512 and that the width adaptation happens downstream of this switch.

Maestro will only allow the switch connected to the GIU to be 512, the other one in the network must not allow it.

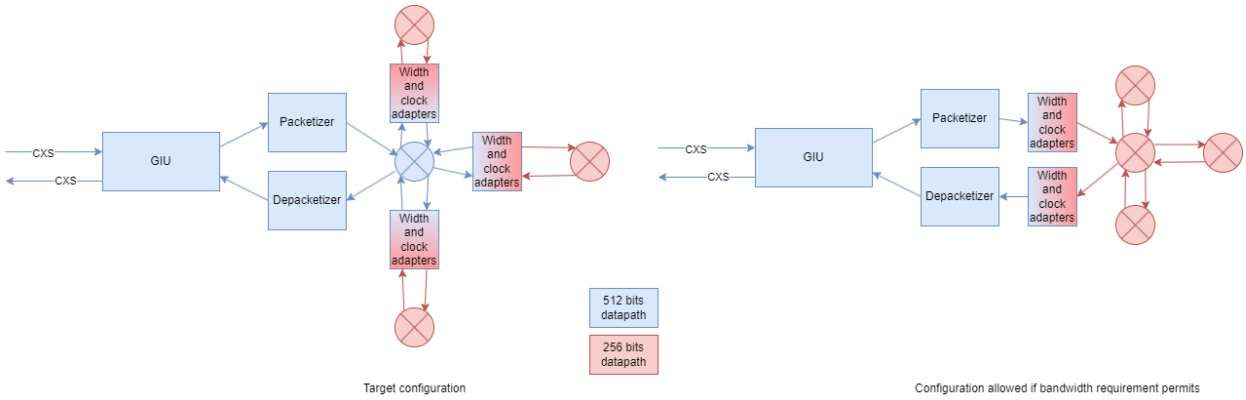


FIGURE 2-3 FIGURE ILLUSTRATING THE WIDTH/CLOCK ADAPTATION BETWEEN THE GIU AND THE REST OF NCORE.

2.4.4 Connectivity map

The CDTI connectivity between different Ncore 3 units shall be optimized by removing any possible unused connectivity by the configuration.

The connectivity optimization between local units shall stay local and shall not take remote chiplets into account. Nevertheless, the connectivity to GIU units may consider system level constraints and shall be conservative and flexible.

The GIU will have 1 SMI port per network it is connected to.

The home DMI and the home DCE are always on the same chiplet, so no message from CN2 (Mrd/Rbr) will ever go to the GIU. Table 2-1 shows the message repartition per network and the mapping to SMI ports. Table 2-2 shows the connectivity between the GIU and the other Ncore units.

	Control Network 0 (CN0) <-> SMI0	Control Network 1 (CN1) <-> SMI1	Control Network 3 (CN3) <-> SMI2	Data Network (DN) <-> SMI3
GIU	○ CmdReq ○ StrReq ○ SnpReq ○ UpdReq ○ SysReq	○ CmdRsp ○ StrRsp ○ SnpRsp ○ UpdRsp ○ SysRsp	○ DtwRsp ○ DtwDbgRsp ○ DtrRsp	○ DtwReq ○ DtwDbgReq ○ DtrReq

TABLE 2-1 MESSAGE REPARTITION PER NETWORK AND NETWORK SMI MAPPING.

	CAIU	NCAIU	DCE	DII	DMI	DVE	GIU
GIU	CN0 CN1 CN3 DN	CN0 CN1 CN3 DN	CN0 CN1	CN0 CN1 CN3 DN	CN0 CN1 CN3 DN	CN0 CN1 CN3 DN	CN0 CN1 CN3 DN

TABLE 2-2 CONNECTIVITY AMONG GIU AND OTHER NCORE UNITS (CONNECTIONS ARE BIDIRECTIONAL)

2.4.4.1 Removing connectivity

Some DII might be accessed only by local target, user might be able to remove connections between the GIU and such DII. It is, however, not recommended.

It is not allowed to remove the connection between the system DII and the GIU.

If Maestro finds that the interleaving scheme of the DMI/DCE/GIU makes it such that some DMI/DCE may never access some of the GIU for every assembly tested Maestro should propose the optimization to the customer, but it is not advisable to do it automatically as it could hinder reusability of the chiplet in future assemblies (although the current specification does not allow it).

2.5 GIU Grouping

2.5.1 GIU Route Group (RG)

A new concept of Route Group (RG) is introduced to allow for the global routing of transaction. RGs contain every GIU that are connecting to the same chiplet, regardless of any address map or interleaving definition.

Similarly, to interleaving groups, multiple sets of RGs can be created at design time to allow for multiple assemblies. Figure 2-4 shows two possible assemblies using the same chiplet. The RGs are represented in blue.

In the example of Figure 2-4 each assembly requires 2 RG sets to be created.

The total number of RG set is a maximum of 8.

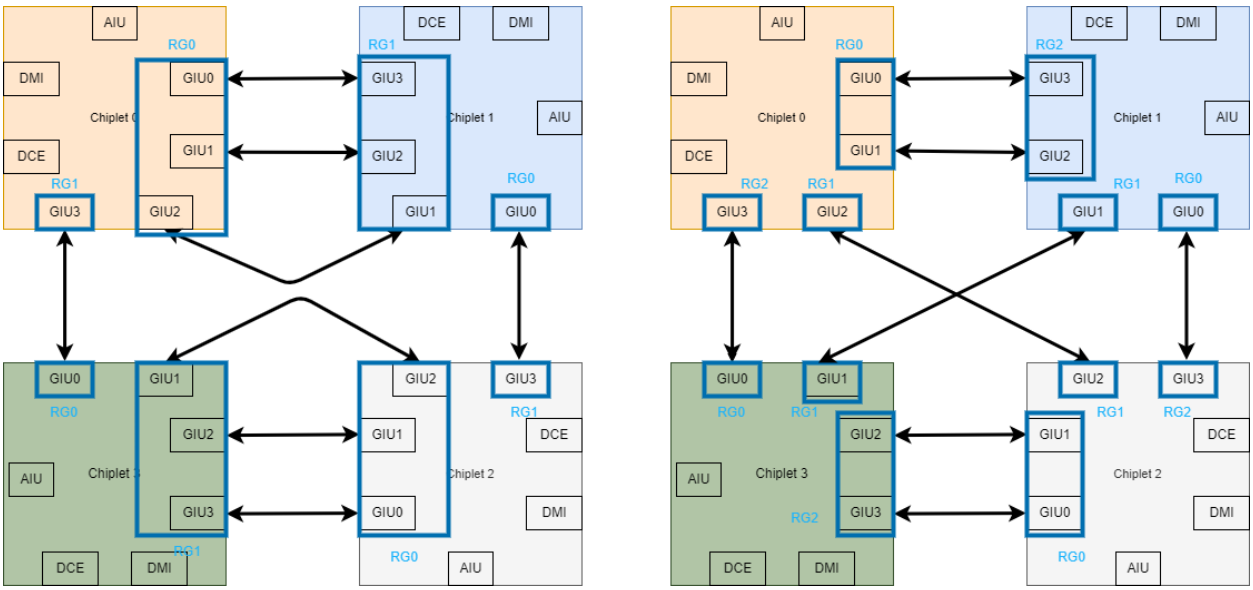
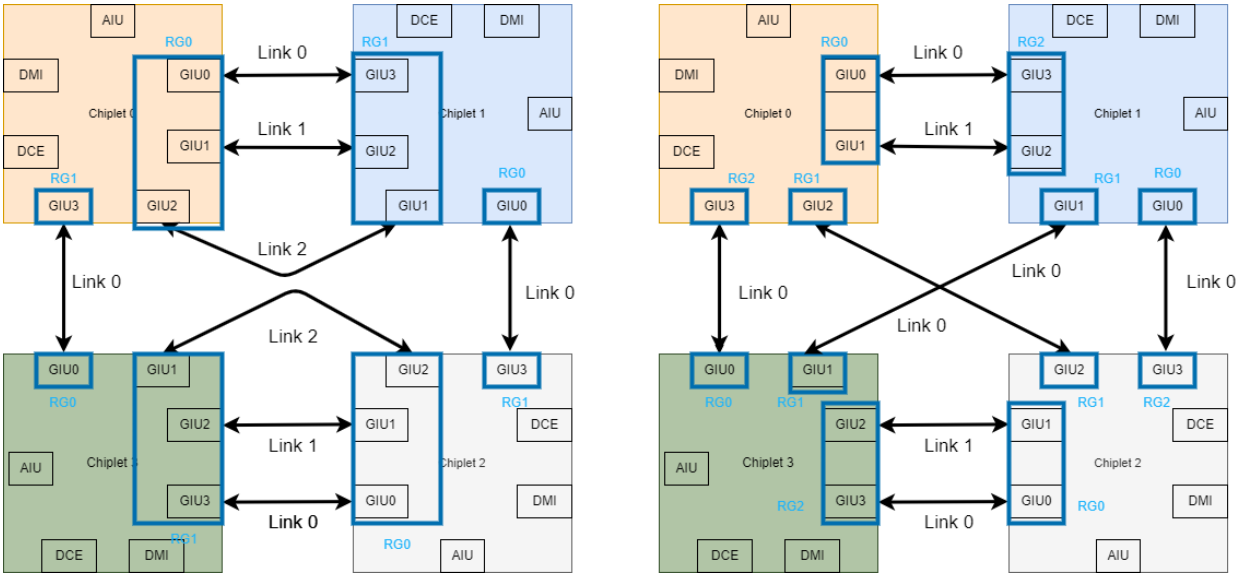


FIGURE 2-4 ILLUSTRATION OF RGS IN TWO DIFFERENT ASSEMBLIES WITH DIFFERENT CONNECTIVITY

2.5.2 LinkId

For every assembly, each D2D link is assigned a LinkId which will allow for the routing of the transaction at system level. The LinkId indexes the link within an RG. Contrary to RGs, each assembly has one and only one LinkId definition.



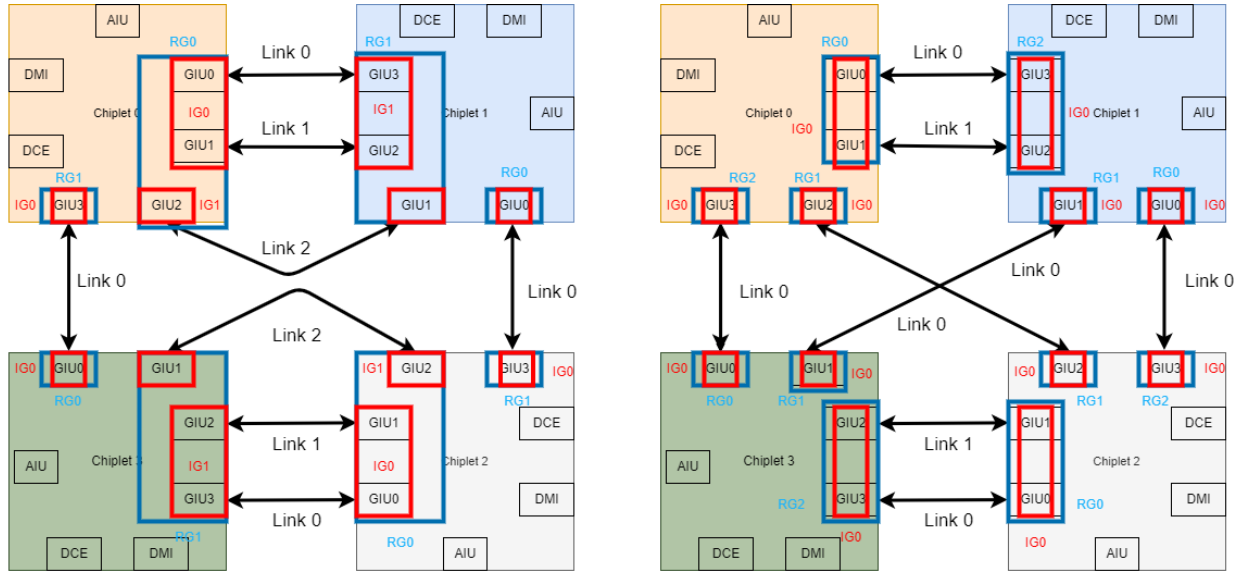
213 FIGURE 2-5 DIFFERENT LINKID ASSIGNMENTS IN TWO DIFFERENT ASSEMBLIES

2.5.3 GIU interleaving group (IG)

216

GIU interleaving group allows for grouping of GIU within the RG. Such a group is called a GIU interleaving group (GIU IG). GIU IGs will be indexed with respect to their RG. Figure 2-6 shows an example of GIU IG of the examples
219 used in Figure 2-4 and Figure 2-5. Interleaving groups are represented in red.

The left assembly splits the group of three GIUs into a 2-way interleaved group and a GIU by itself. In the right assembly, there are no more sub-divisions and GIU RG and GIU IG contain the same GIUs.



222

FIGURE 2-6 IGS' ASSIGNMENTS IN TWO DIFFERENT ASSEMBLIES WITH DIFFERENT CONNECTIVITY

225 Interleaving a GIU will use a similar mechanism as DMI interleaving (multiple interleaving set, predefined interleaving groups and two interleaving functions per associativity) with the following restrictions:

- GIUs which connect to different chiplets cannot be in the same interleaving group.
228 - Up to 8 sets can be created to support multiple assemblies.
- The interleaving functions across the assembly must be identical for all address space.
- The size of each interleaving group can be 1, 2 or 4.

231 GIUs will use their own set of registers for interleaving definition and will exist in every unit which instantiates an address map:

Bits	Field Name	SW Access	HW Access	Reset Value	Descriptions
1	valid	RW	RO	x0	Memory Interleave Group Set valid, when set the GIUIGS specified in AGIUIGS field is being used for DMI interleaving else GIUIGS 0 is used by default.
5:1	AGIUIGS	RW	RO	x0	Active GIU Interleave Group Set in use currently when Valid is set
31:6	Reserved	RO	RO	x0	Reserved

234 TABLE 2-3 XAGIUIGR: ACTIVE GIU INTERLEAVING REGISTER OFFSET : 0X4C8

Bits	Field Name	SW Access	HW Access	Reset Value	Descriptions
2:0	GIUIG2AIFId	RW	RO	x0	Active function ID for 2-way interleaving. Selects the function ID index to be used for interleaving of GIU within the selected GIUIGS.
3	Reserved	RO	RO	x0	Reserved
6:4	GIUIG3AIFId	RW	RO	x0	Active function ID for 3-way interleaving. Selects the function ID index to be used for interleaving of GIU within the selected GIUIGS.
7:5	Reserved	RO	RO	x0	Reserved
10:8	GIUIG4AIFId	RW	RO	x0	Active function ID for 4-way interleaving. Selects the function ID index to be used for interleaving of GIU within the selected GIUIGS.
31:11	Reserved	RO	RO	x0	Reserved

TABLE 2-4 XGIUIFSR GIU INTERLEAVING FUNCTION SET REGISTER. REGISTER OFFSET : 0X4CC

Note that this specification does not allow to interleave DMI/DCEs and GIUs or to interleave DMI/DCE across multiple chiplet. Also, this interleaving mechanism is intended to balance traffic for transaction going to the same chiplet not to simplify the address map for transaction going to different chiplets.

More details regarding the use of GIU RG and GIU IG will be given in the description of the GPRAR and the global routing function.

2.5.4 New parameter RemoteLinkInterleavingObject

RemoteLinkInterleavingObject is a new parameter which allows the mapping of an address into a LinkId. It is an array of objects defined such that:

RemoteLinkInterleavingObject [“MyAssemblyId

“][<MyAssemblyIdIndex>][“MyChipletId”][<MyChipletId>][“RemoteChipletId”][<RemoteChipletId>][“IGId”][<IG>][“IFAddr”][InterleavingFunction(Addr)] = LinkId

The <> means the value of, while strings are required to access the object. This was modified from the multidimensional array because Maestro does not support multidimensional arrays.

For example, to obtain the LinkId of a GIU in the interleaving group 1 from Chiplet 0 to Chiplet 1 one would use the following, assuming there are 4 GIU 2 way interleaved and the interleaving function is on address bit 21:

RemoteLinkInterleavingObject[“MyAssemblyId”][0][“MyChipletId”][0][“RemoteChipletId”][1][“IgId”][1][“IFAddr”][Addr[21]]

MyAssemblyId and MyChipletId are inputs to Ncore, the remote ChipletId comes from the HUI of the GPRAR (see section 2.7)

2.6 Address map

2.6.1 Ncore register space NRS

Each partition shall have its Ncore register region NRR.

NRR shall be extended adding register pages (RP) for GIU per Table 2-5 below. The census register (NRUCR) is extended to provide the number of GIUs per Table 2-6 below. Software may use the fields of the census register to determine which register page numbers are defined and which are reserved.

The offset of the NRS region will depend on the ChipletId and will be calculated as Offset+(ChipletId x Size).

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 Coherence 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	GIU (Gateway Unit Interface)	RPN(nCAIU + nNCAIU + nDCE + nDMI + nDII) to RPN(nCAIU + nNCAIU + nDCE + nDMI + nDII + nGUI - 1)
7	DVE (Distributed Virtualization Engine)	RPN(nCAIU + nNCAIU + nDCE + nDMI + nDII + nGUI)
8	GRB	0xFF (255)

270 TABLE 2-5 RP ASSIGNMENT TO NCORE COMPONENTS

Bits	Field Name	SW Access	HW Access	Description
7:0	nAIUs	RO	RO	Number of AIUs within this NRR.
13:8	nDCEs	RO	RO	Number of DCEs within this NRR.
19:14	nDMIs	RO	RO	Number of DMIs within this NRR.
25:20	nDIIs	RO	RO	Number of DIIs within this NRR.
26:26	nDVEs	RO	RO	Number of DVEs within this NRR.
29:27	nGIUs	RO	RO	Number of GIUs within this NRR.
31:30	Reserved	RO	RO	Reserved (RAZ/WI)

TABLE 2-6 NRR UNIT CENSUS REGISTER (NRRUCR)

273

2.6.2 Boot region

276 Each individual chiplet will have a boot region accessing one of its local DMI/DII. The boot region can **never** be accessed from a remote chiplet so it can be the same address offset in every chiplet.

2.6.3 General purpose address page

Two options were studied:

- The first option is to keep a flat address map. The region can be mapped to either normal memory (DMI) or to device (DII). The DMIs and DIIs can be local or remote and the address decode logic in the AIU directly provide the FUnitId of the local target which is the final destination if it is a local DMI/DII or a GIU if it is a remote DMI/DII.
- The second option is to support a hierarchical address decode. The region can be mapped to either local DMIs and DIIs or to a remote chiplet. Every time a command leaves a GIU the address is decoded again until the destination is reached. **This option was chosen because it helps with scalability.**

A region which is in a remote chiplet can be programmed in two modes:

- Direct assignment of a Link.
- Assignment to a GIU interleaving group.

In direct assignment the interleaving will be ignored.

The regions which are assigned to remote chiplets may be noncontiguous. Then, several regions shall be mapped to the same GIU representing several DIIs or DMIs. The granularity of the regions shall remain unchanged from a monolithic Ncore 3 system.

Each remote region can be flagged as a remote DII region when it is known that the target will be a DII for this specific address range. It can also be flagged as a general-purpose remote region for which the destination can be a DCE, a DMI or a DII.

This distinction allows for some optimization when PCIe traffic crosses chiplet boundaries.

The programming of local DII/DMI region is unchanged from Ncore3.7.

The GIU will contain the same address map as AIUs in the same chiplet and will do the decoding for the commands it injects back into Ncore.

It is allowed for a memory region to be private to a die, to do so one needs to configure the region as local in the die where it belongs and not have it configured as remote in other dies. It is allowed to have a range of address being private in every die (see the local only IO region in the table below).

Figure 2-7 shows an example of an address map spanning across 4 chiplets. Each region's color indicates in which chiplet the target is located and using the color code from the previous figures i.e., orange for chiplet 0, blue for chiplet 1, grey for chiplet 2 and green for chiplet 3. Each column represents the address space as seen by the chiplet labeled at the top of the column. Each chiplet boots independently from a region starting at the same address in every chiplet. This boot region cannot be accessed remotely.

Then each configuration space gets a separate region and are offset based on the ChipletId. This means that the csr offset will depend on the ChipletId.

- Between 500MB and 1GB we can see a region for local only IOs. Those regions cannot be accessed remotely with this address map.
- Between 1GB and 2GB there are IO regions which can be accessed remotely.
- Finally, between 2GB and 4GB there is a memory region which can also be accessed remotely.

It is important to notice that whether a region can be accessed remotely or not depends on the address map setup.

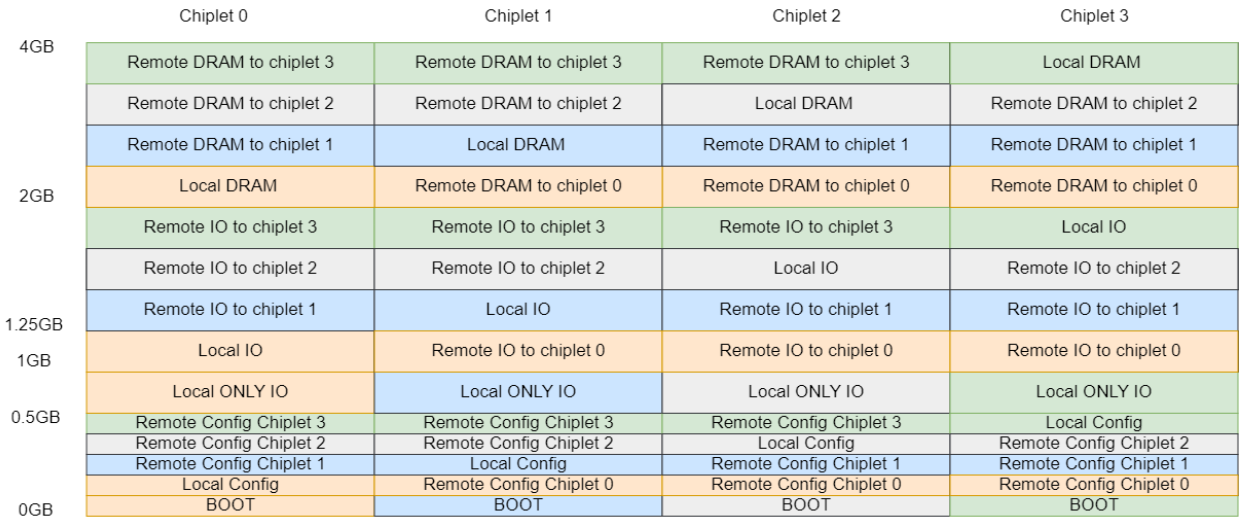


FIGURE 2-7 EXAMPLE OF AN ADDRESS MAP SPANNING ACROSS 4GB.

2.6.4 General guidelines to program the GPRAR

For the address map to be valid the following must not happen :

- An address region can't be programmed as remote to die i in chiplet j and to die j in chiplet i.
- An address region must be configured as local in at least one of the die.

2.7 GPRAR update

2.7.1 GPRAR update for CAIU and IOAIU

2.7.1.1 HUT: Home Unit Type

The HUT field is redefined and still uses bits 29 and 30. Bit 29 will indicate D2D, and the field should be decoded as follows:

- 00: Local system Memory (DMI)
- 10: Local I/O (DII)
- 01: Remote (D2D)
- 11: Remote I/O (D2D_DII)

2.7.1.2 HUI: Home Unit Identifier

The HUI should be interpreted as follows based on the values of HUT.

When HUT == 2'b00, the HUI field is interpreted as the DMI interleaving group Id
When HUT == 2'b10, the HUI field is interpreted as the DII NunitId
When HUT == 2'bx1, the HUI field is interpreted as the ChipletId

2.7.1.3 Interleaving group ID and LinkId

Interleave and LinkId are two new fields and will use bit 19 for Interleave and bits 17-18 for LinkId. It is only applicable when the HUT is x1. If Interleave is set to 1 the LinkId field should be interpreted as a GIU IG, whereas if it is set to 0 it is interpretated directly as a LinkId.

Figure 2-8 shows an example to achieve the address map shown in Figure 2-7 assuming the left assembly is shown in Figure 2-6. It shows how to program those new fields to achieve the address map described above. Similarly, the color of each entry represents where the target is and follows the same color code.

Chiplet 0

	V	HUT	Interleaving	Size	LinkId	HUI	Addr Offset
GPRAR0	1	DII	x	17	x	DII_NUnitId	2000_0000
GPRAR1	1	DII	x	16	x	DII_NUnitId	4000_0000
GPRAR2	1	C2C_DII	0	16	0	Chiplet 1	5000_0000
GPRAR3	1	C2C_DII	0	16	2	Chiplet 2	6000_0000
GPRAR4	1	C2C_DII	0	16	0	Chiplet 3	7000_0000
GPRAR5	1	DMI	x	17	x	DMI IG	8000_0000
GPRAR6	1	C2C	1	17	IG0	Chiplet 1	a000_0000
GPRAR7	1	C2C	1	17	IG1	Chiplet 2	c000_0000
GPRAR8	1	C2C	1	17	IG0	Chiplet 3	e000_0000

Chiplet 1

V	HUT	Interleaving	Size	LinkId	HUI	Addr Offset
1	DII	x	17	x	DII_NUnitId	2000_0000
1	C2C_DII	0	16	0	Chiplet 0	4000_0000
1	DII	x	16	x	DII_NUnitId	5000_0000
1	C2C_DII	0	16	0	Chiplet 2	6000_0000
1	C2C_DII	0	16	2	Chiplet 3	7000_0000
1	C2C	1	17	IG1	Chiplet 0	8000_0000
1	DMI	1	17	x	DMI IG	a000_0000
1	C2C	1	17	IG0	Chiplet 2	c000_0000
1	C2C	1	17	IG0	Chiplet 3	e000_0000

Chiplet 2

	V	HUT	Interleaving	Size	LinkId	HUI	Addr Offset
GPRAR0	1	DII	x	17	x	DII_NUnitId	2000_0000
GPRAR1	1	C2C_DII	0	16	2	Chiplet 0	4000_0000
GPRAR2	1	C2C_DII	0	16	0	Chiplet 1	5000_0000
GPRAR3	1	DII	x	16	x	DII_NUnitId	6000_0000
GPRAR4	1	C2C_DII	0	16	0	Chiplet 3	7000_0000
GPRAR5	1	C2C	1	17	x	Chiplet 0	8000_0000
GPRAR6	1	C2C	1	17	x	Chiplet 1	a000_0000
GPRAR7	1	DMI	1	17	x	DMI IG	c000_0000
GPRAR8	1	C2C	1	17	x	Chiplet 3	e000_0000

Chiplet 3

V	HUT	Interleaving	Size	LinkId	HUI	Addr Offset
1	DII	x	17	x	DII_NUnitId	2000_0000
1	R_DII	0	16	0	Chiplet 0	4000_0000
1	C2C_DII	0	16	2	Chiplet 1	5000_0000
1	C2C_DII	0	16	0	Chiplet 2	6000_0000
1	DII	x	16	x	DII_NUnitId	7000_0000
1	C2C	1	17	IG0	Chiplet 0	8000_0000
1	C2C	1	17	IG1	Chiplet 1	a000_0000
1	C2C	1	17	IG0	Chiplet 2	c000_0000
1	DMI	1	17	x	DMI IG	e000_0000

FIGURE 2-8 EXAMPLE OF HOW TO PROGRAM GPRAR

357

2.7.1.4 Comparisons between the existing and proposed changes in GPRAR

NCAIU GPRAR$_x$: General Purpose Region Attribute Register [Offset: 0x0400]				
Bit	Name	Description	Access	Reset
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	Determines the setting of our internal OR [1:0] field which defines the behavior of the agent at the bottom of Ncore.	RW	0x0
5	NC	Non-Coherent - this bit applies only for AXI transactions, ignore for CHI/ACE/ACE-Lite and use transaction property NC: 0: Use coherent mode as indicated by incoming transaction 1: Enforce non-coherent transaction	RW	0x0
7:6	NSX	This field describes the Security Level required to access this region. NSX[0] is mapped to NS(non-secure) NSX[1] is mapped to NSE (non-secure extension, reserved for later use)	RW	0x0
8	Rsvd	Reserved	RZWI	0x0
13:9	HUI	When HUT == 2'b00, the HUI field is interpreted as the DMI interleaving group Id When HUT == 2'b10, the HUI field is interpreted as the DII NunitId When HUT == 2'bx1, the HUI field is interpreted as the ChipletId	RW	0x0
16:14	Rsvd	Reserved	RZWI	0x0
18:17	LinkId	It is only applicable when the HUT is 2'bx1. When Interleave bit is set as 1'b1, this field is interpreted as a GIU IG. It is interpreted as a LinkId when the interleave bit is 1'b0.	RW	0x0
19	Interleave	When this bit is set to 1'b1, LinkId field holds the GIU interleaving group definition	RW	0x0
25:20	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
28:26	Rsvd	Reserved	RZWI	0x0
30:29	HUT	This field indicates the Home Unit Type: • 00: Local system Memory (DMI) • 10: Local I/O (DII) • 01: Remote (D2D) • 11: Remote I/O (D2D_DII) – to enforce the order such as PCIe	RW	0x0
31	Valid	This bit indicates if the region is valid. 0: Invalid mapping 1: Valid region mapping	RW	0x0

360

TABLE 2-7 GPRAR DEFINITIONS IN NCORE 3.8 WITH UPDATES

NCAIU GPRAR$_x$: General Purpose Region Attribute Register [Offset: 0x0400]				
Bit	Name	Description	Access	Reset
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	Determines the setting of our internal OR [1:0] field which defines the behavior of the agent at the bottom of Ncore.	RW	0x0
5	NC	Non-Coherent - this bit applies only for AXI transactions, ignore for CHI/ACE/ACE-Lite and use transaction property NC: 0: Use coherent mode as indicated by incoming transaction 1: Enforce non-coherent transaction	RW	0x0
7:6	NSX	This field describes the Security Level required to access this region. NSX[0] is mapped to NS(non-secure) NSX[1] is mapped to NSE (non-secure extension, reserved for later use)	RW	0x0
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 **Chip-to-Chip:** Gateway Interleave Group	RW	0x0
19:14	Rsvd	Reserved	RZWI	0x0
25:20	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
28:26	Rsvd	Reserved	RZWI	0x0
30:29	HUT	This field indicates the Home Unit Type: 00: System Memory 01: Hierarchical Gateway Interface 10: Peripheral 11: Chip-to-Chip interface Note, the pattern has been chosen to retain compatibility with the current, 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

363

TABLE 2-8 GPRAR DEFINITIONS IN NCORE 3.7 FOR REFERENCE

2.7.2 GPRAR update for DCE

366 ### *2.7.2.1 HUT: Home Unit Type*

The HUT field update is the same as for a CAIU or an IOAIU as described in section 2.7.2.1.

2.7.2.2 HUI: Home Unit Identifier

The HUI field update is the same as for a CAIU or an IOAIU as described in section 2.7.1.2.

2.7.2.3 Interleaving group ID and LinkId

These two subfields have the same usage assumptions and definitions in an IOAIU or a CAIU as described in section 2.7.1.3.

2.7.3 GPRAR in the GIU

An address map lookup is required in the GIU in order to find the local destination of a CmdReq. It only needs to look up the address map when a request has reached its destination die (the ChipletId of the CmdReq TargetId matches the current ChipletId) to find which local unit it needs to send the request to.

Although only local targets are decoded from the address, the entire address map is programmed inside the GIU.

2.8 Ncore CSTI

The Control and Status Transport Interconnect network is dedicated to access configuration, status, and error registers in Ncore 3.

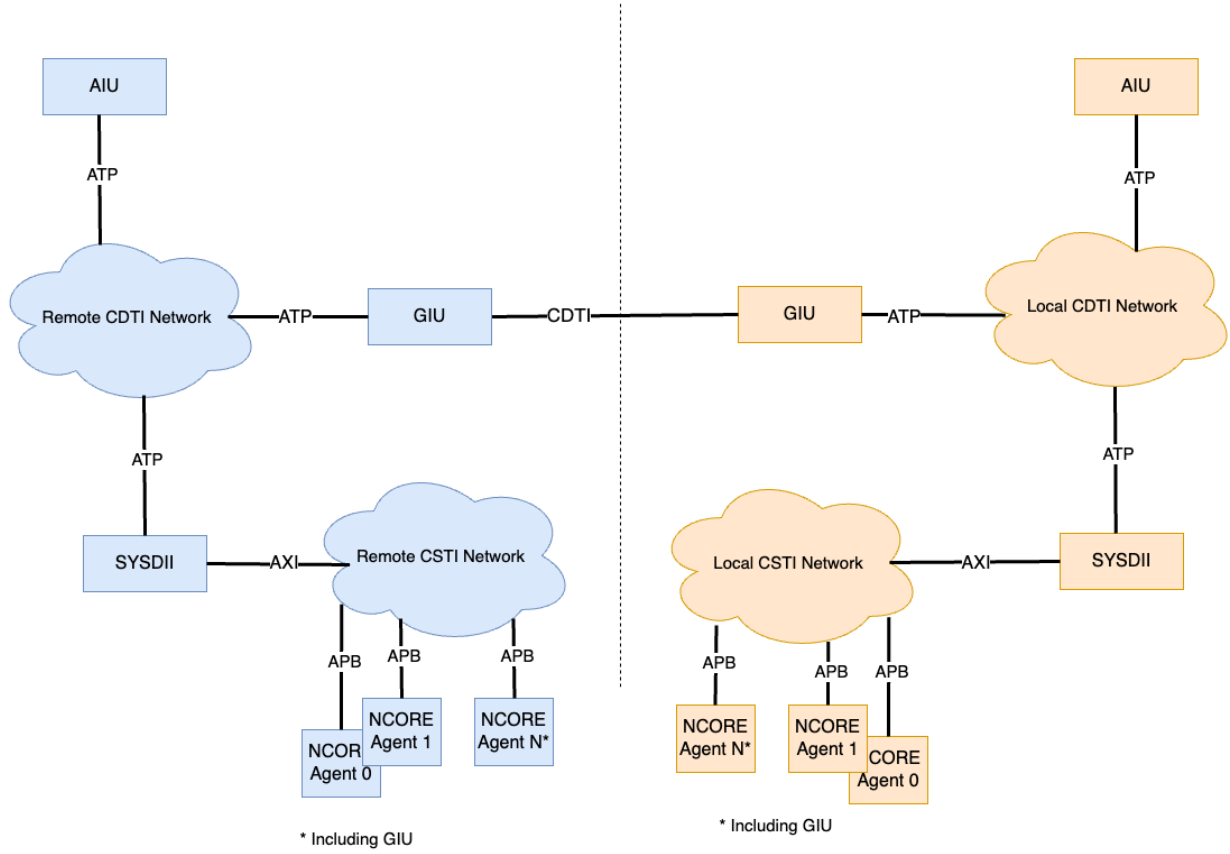
There are several entry points to the CSTI network:

- SYSDII (which is a gateway with the CDTI network)
- Debug port featuring an APB interface.

Each chiplet shall have its own debug port which can only be used to access the registers in the same partition of Ncore.

The CSTI of the remote chiplet can only be reached from an AIU via the CDTI network to a remote SYSDII. In that case the message uses the existing D2D channels from CDTI network.

Each chiplet must contain a SYSDII and an APB debug port which are responsible for access to the Ncore register on their partition of Ncore.



393

FIGURE 2-9 CONFIGURATION TO A REMOTE DIE/CHIPLET

396 # 2.9 Namespace

2.9.1 Global FUnitId

399 Each unit in the system including the GIU is assigned a global FUnitId which is a concatenation of a ChipletId and a local FUnitId. This Id is unique in the entire assembly and will be used to direct message to the correct destination.

402 Each Ncore unit must be provided with their local FUnitId and ChipletId as tie offs.

The ChipletId will be an input to the gen_wrapper, and it will be the customers' responsibility to provide the correct value depending on what assembly the chiplet is used in.

405

2.9.2 Logical processors (used for exclusive access)

For exclusive monitor, the global FUnitId of the sender must be used instead of the local FUnitId. This also implies in the case of DCE exclusive monitor that one monitor per logical processor in the biggest assembly must be created. The mapping between the monitor and the logical processor will use the same mechanism as the coherency agents.

2.9.3 Stashing agents

Stashing is not supported across chiplet boundary. The list of stash capable agents is unchanged compared to Ncore 3.6/3.7. The details of the behavior of the GIU when receiving a stash transaction will be detailed in the protocol section of this specification.

2.10 Physical and link layer between two chiplets

This specification does not make any assumption on the physical layer, or the link layer used to carry flits from one chiplet to the other. As such, any controller which provides a CXS.B interface which is compatible with the one described later in this specification can be used.

Compatibility with common controllers using UCIe and PCIe PHY will be given in a separate section. The use of a controller not explicitly defined in this specification may or may not be compatible with the GIU and will need to be checked.

In general, any controller whose CXS interface has properties compatible with the one described in this specification and is used strictly to stream (i.e., it does not make any assumption on the CXS flit) should be compatible.

2.11 Tunnel

2.11.1 Message classes

The tunnel:

- shall provide independent transmission channels ensuring independent progress between message classes.
  - using separate credits for each message class
  - providing independent flow controls for each message class
- shall provide a shared link for all message classes (VC0: CN0, VC1: CN1, VC2: CN3, VC3: DN)
  - mapping the SMI networks CN0 CN1 CN3 DN.
  - adding a new message class SYST for system messages for miscellaneous purposes (initialization sequence, credit exchange, …)
- shall provide fairness between message classes (fairness in VC selection but also fairness in credits return)
- shall guarantee delivery without duplication or dropping and ordering requirement per message class.
- shall provide enough credits to match the bandwidth requirements in function of the round-trip latency.
- shall reallocate unused resources (credits) in absence of remote chiplet.
- Shall consider some miscellaneous handshake and parameter exchange messages to and from 3rd party controller and PHY.

In summary, virtual channels with separate buffer resources will be used to guarantee that no dependency is created between different message classes.

2.11.2 Packing/Unpacking

The packet algorithm used to send the payload on the CXS interface is the same as CHI-C2C with the following considerations:

- The granule size is 10 Bytes.
- The meaning of the protocol header is updated as shown in Figure 2-10.
- The granules are organized in four groups G0-G5, G6-G11, G12-G17, G18-G23.
- This packet format is compatible with CHI-C2C Format X.
- This packet format is compatible with UCIe 256 Byte Latency Optimized stream and 68 Byte Stream formats.

More specifically the rules for packing are as follows:

- A message can start in any granule.
  - It has to start the very beginning of a new granule.
- Multi-granule messages must use contiguous granules.

- Multi-granule messages can span across multiple containers, in which case the must continue in G0 of the next container.
    - One container is 256 Byte.
- An empty container is allowed.
- The unused bit must be 0. This includes padding bits in granules and unused granules.
- The data messages are passed in the same order as they were received from Concerto.
- Granules are organized into four groups such that:
    - If a granule within the group is empty all the subsequent ones also are.
    - No more than 4 messages of the same type per 64 bytes are allowed (applies to response message which only occupy one granule).

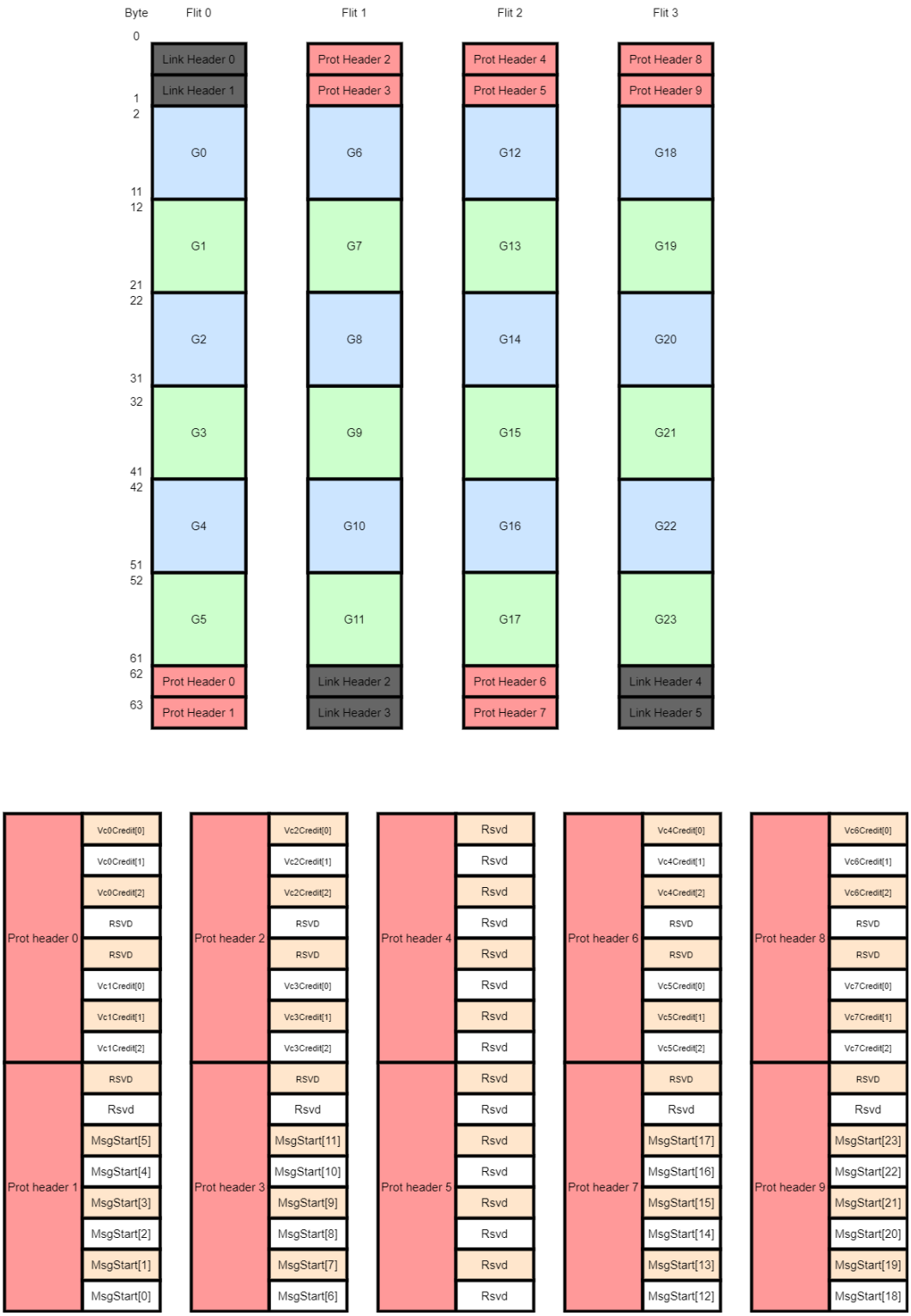


FIGURE 2-10 PACKET FORMAT USED BY THE GIU

2.11.3 Message formats

The Concerto parameters and the value they are assigned to for D2D system is described in The Ncore 3.8 parameter specification.

VC name	Number of granules
VC0	3
VC1	2
VC2	1
VC3	9

TABLE 2-9 MESSAGE SIZE FOR EACH VC

2.11.4 CXS flits compatibility for Synopsys UCIe controller

2.11.4.1 32B streaming (Not implemented and tested for Ncore 3.8.0)

This mode is to be used when the CXS.B interface is 32B. It will NOT be implemented in Ncore 3.8.0 but is kept in this specification for future reference.

The UCIe flit format will be set to format 2 (68B streaming). In this mode of operation, the GIU must ensure to send two CXS flit without bubbles in between. The controller will not modify the data payload.

The packet that we have defined will take 8 cycles to be transmitted on the CXS.B interface and the flit will need to be sent one by one without any bubble.

The timing diagram below shows the expectation from the Synopsys controller. In this diagram flit1 and flit2 will have the content of flit 1 in our packet definition.

This means that a transfer may not start until a full 64B flit is ready to be dispatched (2 link credits)

Figure 3-4 CXS Flit Transfers for Streaming 32B

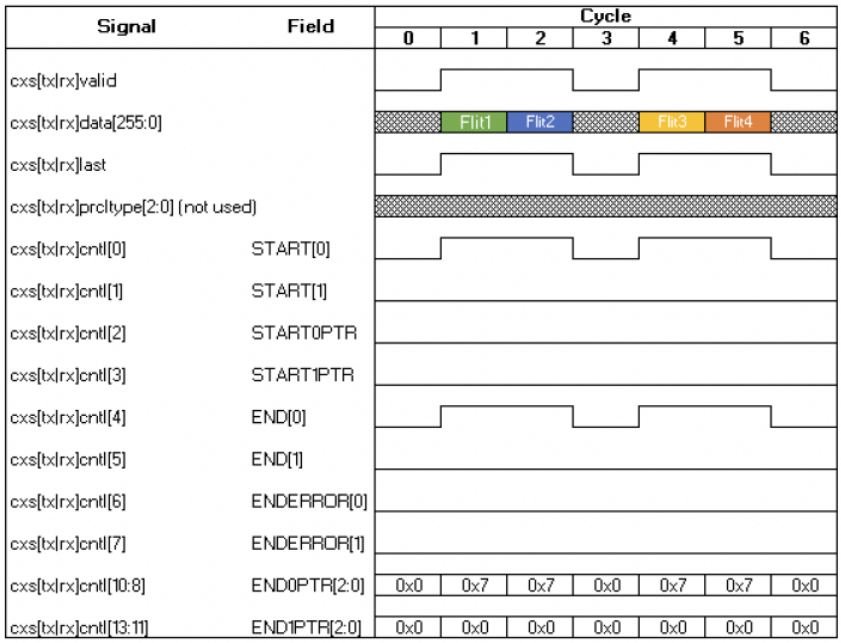


FIGURE 2-11 THE CXS.B INTERFACE WITH 32 BYTE INTERFACE – FORMAT 2

501

2.11.4.2 256B streaming (only mode implemented for Ncore 3.8.0)

504 This mode is to be used with a packet formatted using the 256B latency optimized (Format 6) with optional bit UCIe format. Synopsys controller expectation is shown below. In this diagram, flit1 corresponds to the entire packet defined earlier (256B in 4 64B chunk). Bubbles are only allowed at 256B boundaries meaning that the 507 transfer can only start when the entire packet is ready to be sent (4 link credits). This means that the credit handling is always exchanged in a granularity of 4 credits, which requires FOUR 64-Byte flits must be transmitted without any gap between the GIU and UCIe controller.

510 Note that a bubble means that the valid is not asserted for a cycle, however it is perfectly allowed to send a valid empty flit if not enough message are ready to be sent.

Figure 3-5 CXS Flit Transfers for Streaming 256B / ARM CHI-C2C

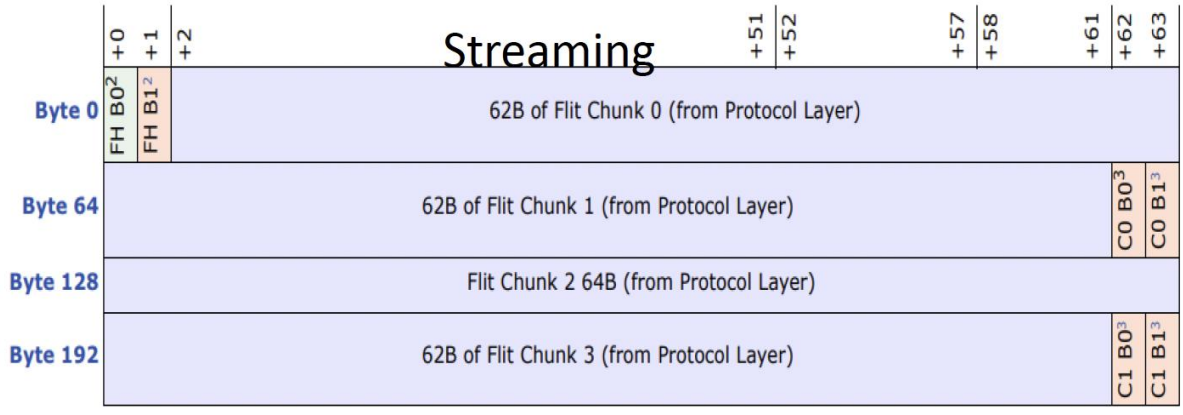
Signal	Field	Cycle 0	1	2	3	4	5	6	7	8	9	10
cxs[tx|rx]valid												
cxs[tx|rx]data[511:0]			Flit1	Flit1	Flit1	Flit1		Flit2	Flit2	Flit2	Flit2	
cxs[tx|rx]last												
cxs[tx|rx]prcltype[2:0] (not used)												
cxs[tx|rx]cntl[0]	START[0]											
cxs[tx|rx]cntl[1]	START[1]											
cxs[tx|rx]cntl[3:2]	START0PTR[1	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
cxs[tx|rx]cntl[5:4]	START0PTR[1	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
cxs[tx|rx]cntl[6]	END[0]											
cxs[tx|rx]cntl[7]	END[1]											
cxs[tx|rx]cntl[8]	ENDERROR[0]											
cxs[tx|rx]cntl[9]	ENDERROR[1]											
cxs[tx|rx]cntl[13:10]	END0PTR[3:0]	0x0	0x0	0xF	0x0	0xF	0x0	0x0	0xF	0x0	0xF	0x0
cxs[tx|rx]cntl[17:14]	END1PTR[3:0]	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

513

FIGURE 2-12 THE CXS.B INTERFACE WITH 64 BYTE INTERFACE – FORMAT 6

516 ### 2.11.4.3 *Implementation suggestion for the CXS.B interface*

Even though both Format 2 (as shown in Figure 2-11) and Format 6 (as shown in Figure 2-12) streaming modes are
519 all supported by Synopsys UCIe controller, Format 6 is chosen by Synopsys to better compatible with the Format X as defined in ARM CHI C2C specification. Figure 2-13 defines the packet layout and please note that application layer needs to drive 2'b00 for byte 0 so the Synopsys UCIe controller can change this field for its transport purpose.



522

FIGURE 2-13 UCIE FORMAT 6 PACKET LAYOUT FOR STREAMING

A GIU needs to be flexible to interface with different data width and different frequencies depending on different data rates and different packages a UCIe interface is chosen to support. Necessary frequency and data width adapters need to be considered for the CXS.B interface to seamlessly interface with Synopsys and other 3rd party UCIe controller IPs.

Please note that the packet format defined in Figure 2-10 is the same as the one used by CHI-C2C Ncore 3.8 but it is **NOT** compatible with CXL or CHI-C2C applications.

2.12 Routing layer

The routing layer is responsible for sending a message to the correct destination base on the TargetId and the InitiatorId of the message.

TargetId

The route field remains local to each chiplet and will be recalculated when crossing chiplet boundaries. However, TargetId and InitiatorId become global fields and will be the concatenation of {LinkId, ChipletId, FUnitId, PortId}. Those fields are not updated when changing chiplet except for CmdReq for which the FUnitId portion is only populated once the CmdReq reaches its destination chiplet.

2.12.1 Packetizer update

2.12.1.1 Route calculation

The packetizer decodes the ChipletId of the TargetId and finds if the destination is remote or local.

If the target is local, the packetizer uses the {FUnitId, PortId} to acquire a route.

If the target is remote the packetizer uses the {ChipletId, LinkId} to acquire a route. A different object will be provided by Maestro for a remote route. Its structure is similar to the existing pathLut object for local routing. Its structure will follow the same format as the local PathLut object and will contain all the possible values of initiator from the same tile, assembly, ChipletId,TargetChipletId and LinkId:

```
GlobalpathLut=[{targ_id : “{ 8’b0, 1’b0, 2’b0, 2’b1,2’b0}”,route: ”8’b00001111”},
               {targ_id : “{ 8’b0, 1’b0, 2’b0, 2’b1,2’b1}”,route:  “8’b11110000”},
               {targ_id : “{ 8’b0, 1’b0, 2’b0, 2’b1,2’b2}”,route:  ”8’b00001111”},
               {targ_id : “{ 8’b0, 1’b0, 2’b0, 2’b1,2’b3}”,route:   ”8’b11110000”},

               {targ_id : “{ 8’b0, 1’b0, 2’b1, 2’b0,2’b0}”,route: ”8’b00110011”},
               {targ_id : “{ 8’b0, 1’b0, 2’b1, 2’b0,2’b1}”,route:” 8’b11001100”},
               {targ_id : “{ 8’b0, 1’b0, 2’b1, 2’b0,2’b2}”,route:” 8’b00110011”},
               {targ_id : “{ 8’b0, 1’b0, 2’b1, 2’b0,2’b3}”,route:” 8’b11001100”}
 ]
```

The above example could describe the Global pathLut object for the unit with FUnitId 0 (8’b0) with only one assembly (1’b0) and two chiplet (2’b0 and 2’b1) and two links (although 4 are contained in the object 2’b0, 2’b1, 2’b2, 2’b3, see section 2.12.1.2 for the rational).

The first four entries in the array describe the path for message to go to chiplet 1 from chiplet 0. There are two links so two different routes and if the message contained LinkId 0 or 2, it maps to route 8’b00001111 while LinkId1 and LinkId 3 it maps to route 8’b11110000.

The next four entries in the array describe the path for message to go to chiplet 0 from chiplet 1. There are two links so two different routes and if the message contained LinkId 0 or 2, it maps to route 8’b00110011 while LinkId 1 and LinkId 3 it maps to route 8’b11001100.

2.12.1.2 LinkId overflow

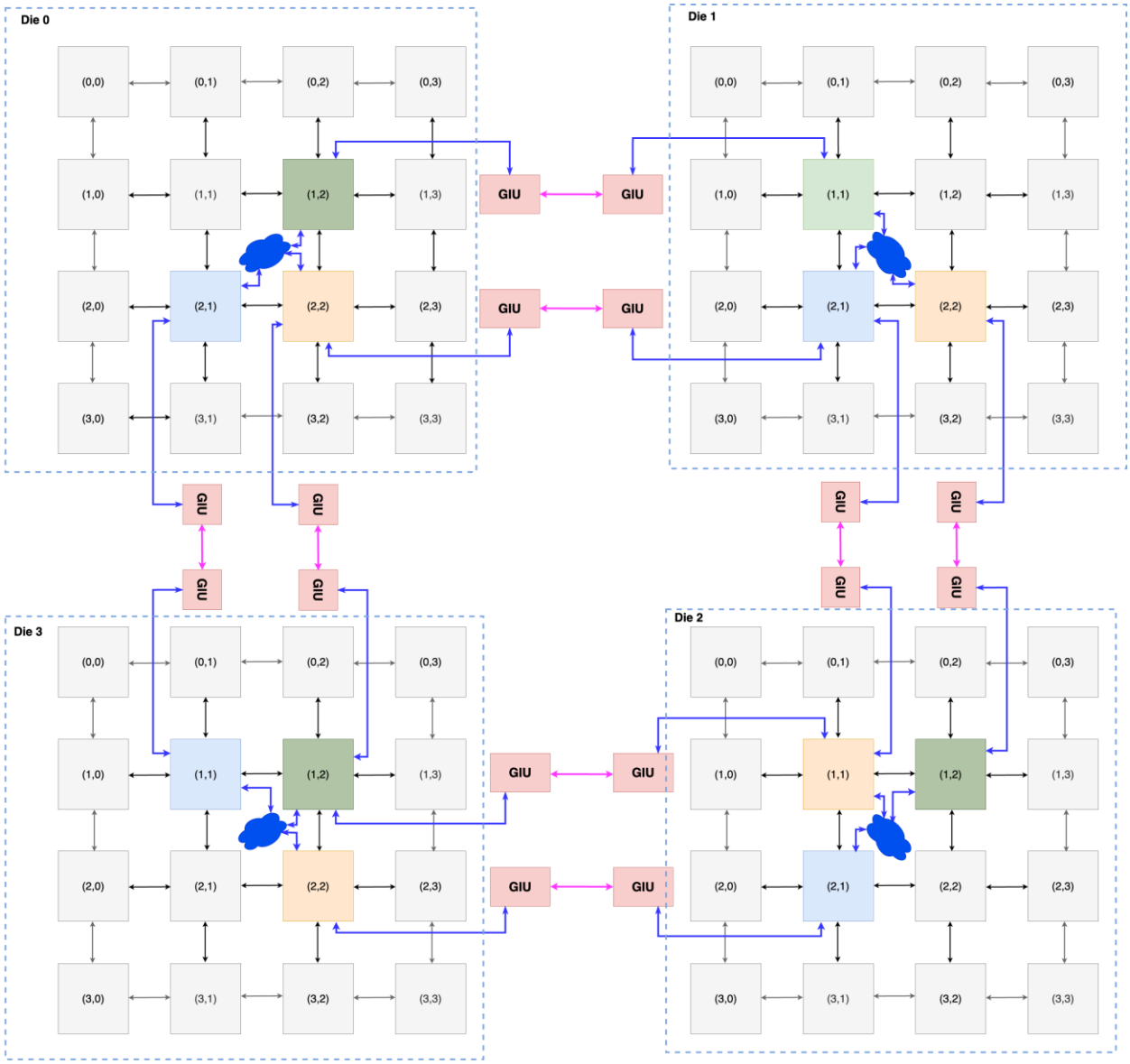
It is possible for the packetizer to decode a LinkId which does not exist. This would typically happen when the number of Links between chiplet is not homogeneous between chiplets. In this case, a modulo will be applied. For example, if the TargetId contains a LinkId of 3 (respectively 2) but there are only 2 links, it will send the request to link 1 (respectively 0). Maestro will provide a route in the globalPathLut for every value possible of the incoming LinkId (4) to remove complexity from the hardware.

2.12.1.3 Tiling considerations

In order to be able to tile the GIU we will not be able to use the InitiatorId of the packet anymore. Instead, the FUnitId will come to the packetizer as a tie-off.

2.12.2 Hierarchical routing and deadlock considerations for an assembly

588 A hierarchical routing methodology [1] is taken to divide and conquer inter-die and intra-die network constructions, their corresponding deadlock checking and to implement deadlock avoidances accordingly.



591 FIGURE 2-14 THE INTEGRATED VIEW OF THE HIERARCHICAL ROUTING FOR AN ASSEMBLY

An example assembly consists of 4 dies of 4x4 mesh is shown in Figure 2-14. For each die, the nodes shown in gray
594 color are only connected to the local Ncore interconnect works. The nodes shown in other colors are called boundary nodes because they carry ONLY global traffic between and among dies in an assembly. The global

networks are separate networks, they are decoupled with the local networks and shown in blue color. Please note that the local networks, which are shown and connected by the black lines, carry both global and local traffic, the global traffic only limit to those global traffic targets to be sank by the local targets or the egress global traffic from the local initiators to the other die(s). The global traffic finally is routed through the boundary nodes on a local die before it can embark on the global routing via the global networks for an assembly.

2.12.2.1 Local networks, corresponding routing and checks

Local networks' construction and corresponding deadlock checks are the same as what has been done for a standalone Ncore instance prior to Ncore 3.8.0.

2.12.2.2 Global networks, corresponding routing and checks

The global networks are the new additions to Ncore 3.8.0 to address the global (inter-die) traffic between and among dies as shown in Figure 2-15. The global networks only include those boundary nodes in each contributing die in an assembly, and they are connected with separate and dedicated networks shown in blue color to simplify the overall networks' constructions, deadlock checking and to implement deadlock avoidance schemes if necessary.

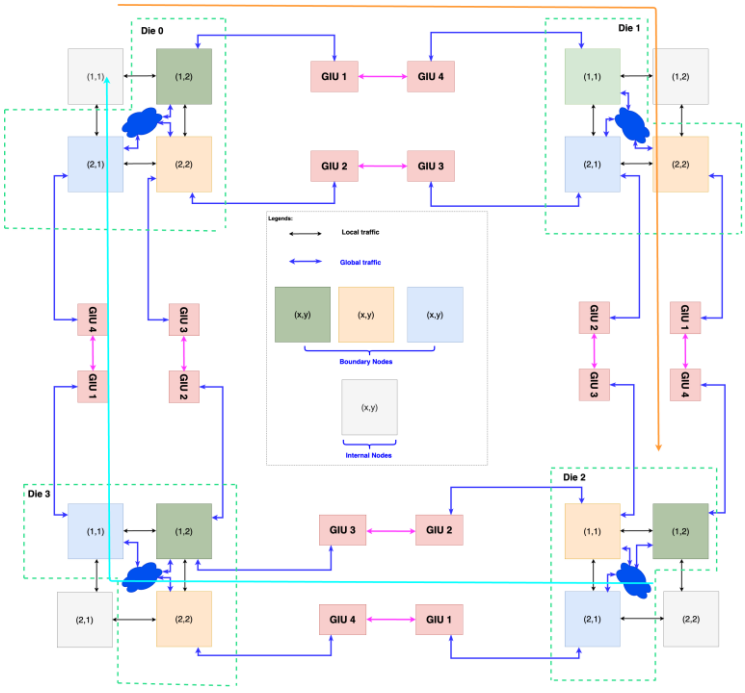


FIGURE 2-15 THE GLOBAL NETWORKS FOR AN ASSEMBLY

Deterministic routing such as X->Y routing must be used in global network constuctions and corresponding deadlock avoidance implementation as shown in orange and cyan lines in Figure 2-15 with extra benefit described next.

2.12.2.3 Handling of integrating a rotation of an existing die

It is a common practice to reuse an existing die and rotate it by 90, 180 or 270 degrees in clockwise to construct an assembly as shown in Figure 2-15. Once a die is rotated by 90 or 270 degrees, a native X->Y routing for a die behave as a Y-> X routing. A loop is built if global and local traffic are not separated and the dies are connected as shown in Figure 2-15. However, it is not a problem for the hierarchical approach taken in Ncore 3.8 as described above since the local traffic always takes the local routes. If it is deadlock free for the original die, it is deadlock free no matter how it is rotated. The local traffic is transparent to a rotation.

However, the global network might require some special handling if an existing die is connected to its rotated version as shown in Figure 2-16.

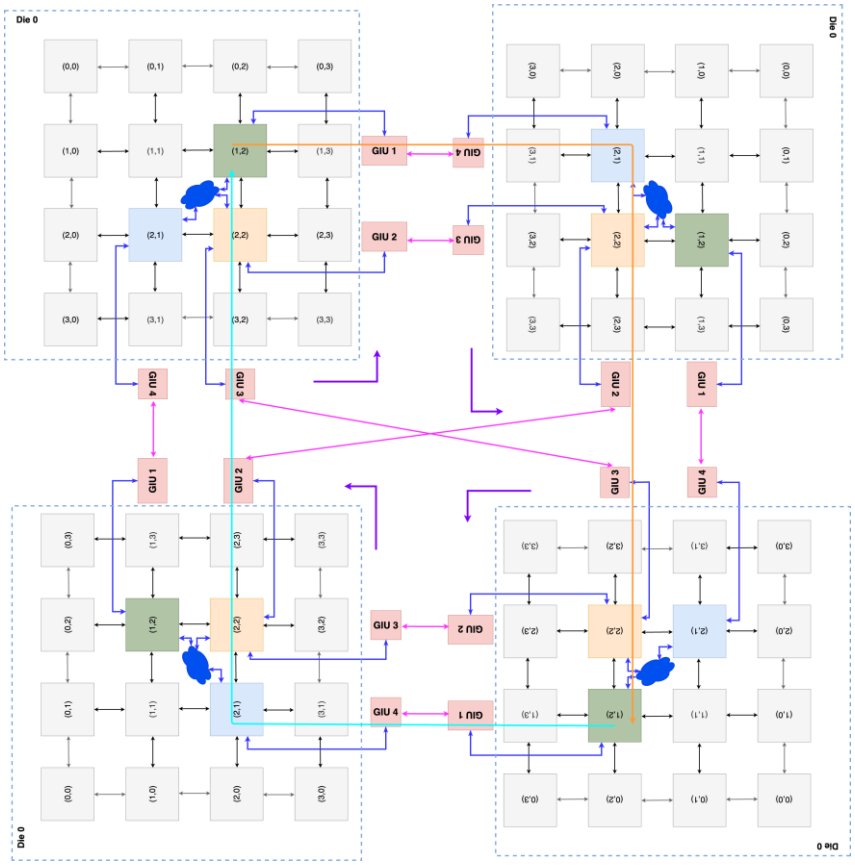


FIGURE 2-16 AN ASSEMBLY CONSISTS OF A DIE AND ITS ROTATED REPLICA

In summary, to guarantee the absence of deadlock the following requirements must be fulfilled:

- Local routes must be analyzed to be deadlock free and verified by Maestro.
- Global networks which is constructed and connected with the boundary nodes from each die must be deadlock free and verified by Maestro

- Each boundary node is a safe node [1] because each GIU is a transport layer entity and does NOT introduce any dependency between its ingress and egress paths
- Global credits have been applied throughout the entire assembly, which guarantees the traffic comes to a GIU(s) can be drained in finite time to avoid deadlocks

For the purpose of this specification in which only 2x2 mesh and fully connected configuration are supported and it must use XY routing for the 2x2 mesh and to the shortest path for fully connected.

3 Protocol Layer

There are no new concepts at the protocol level, instead some existing concepts are updated to match the new constrains brought by chiplet-to-chiplet communication and chiplet reusability.

Specifically, in this chapter we will discuss the update made to the TargetId and InitiatorId fields. Then we will discuss how credit will be handled for remote targets. We will then move on to discuss how snoop filters and basic exclusive monitors will be assigned in the DCE.

Annotated transaction diagrams for selected flows will also be provided in this section.

3.1 InitiatorId and TargetId

As mentioned earlier in the specification the InitiatorId and TargetId are now a concatenation of {LinkId, ChipletId, FUnitId, PortId} which is an update from {FUnitId,PortId} for single die configuration The InitiatorId contains and propagates the same LinkId as the TargetId.

The use of the ChipletId and FUnitId, also called Global FUnitId (see section 2.9.1) identifies uniquely an Ncore Unit.

The addition of the LinkId identifies a unique route through a unique GIU.

PortId will remain unused because of the restriction to 4CN1DN configurations (current Ncore only makes use of PortId in 2CN1DN configurations).

Single die configuration must remain unchanged and continue to use the concatenation of {FUnitId, PortId}.

3.2 TargetId update in the GIU for CmdReq

CmdReq obtain their TargetId from the address map. Consequently, when a transaction is going to a remote target, the FUnitId portion of the TargetId cannot be obtained and the address needs to be decoded again inside the GIU before being put back inside the network with an updated TargetId.

An address which hits a DMI region can be processed coherently in which case it needs to be sent to DCE or non-coherently in which case in needs to be sent directly to DMI. In most cases, decoding the type of CmdReq (CmType) is sufficient to decide if a CmdReq needs to go to a DMI or a DCE. However, for some opcodes (in particular CMOs), it can go to either one of them depending on the native interface attribute (for CHI the SnpAttr). GIU can't make use of this attribute because it does not have access to it.

For this reason, it was decided that the GIU will look at the CH attribute of the CmdReq which was a don't care up until Ncore 3.7. If the CH attribute is set to 1, and the address hits a DMI region, then the transaction will be sent to DCE. If it is set to 0 it will be sent to DMI.

Consequently, AIUs will need to update the logic to derive the CH attribute according to the table in the updated CCMP for Ncore 3.8.

3.3 TargetId update in the GIU for UpdReq

The TargetId of an UpdReq needs to be updated in the GIU similarly to CmdReq except the target is always a DCE.

3.4 LinkId calculation

For CmdReq and UpdReq, the LinkId is obtained from the address map as described and explained in sections 2.6 and 2.7. Message that derives from a CmdReq will reuse the same LinkId.

For SnpReq, the LinkId is obtained from the GIU interleaving object called RemoteLinkInterleavingObject (see section 2.5.4) which DCE will need to have access to. It will apply this interleaving to the address contained in the snoop request. Note that it will require looking at the address map to obtain which IG this address belongs to.

3.5 Protocol credits

3.5.1 Existing Ncore credit

Ncore currently has the following credit types:

- Credit for CmdReq between requester and target (AIUs-> DCE/DMI/DII). This credit pool is managed by the AIUs and the receiver has the buffer required to sink the message. CmdRsp returns the credit.
- Credit for SnpReq between DCE and AIUs. This credit pool is handled by DCE, and the AIUs have the buffer space to sink the message. SnpRsp returns the credit.
- Credit for MrdReq between DCE and DMI. This credit pool is managed by DCEs and the DMIs have enough buffer space to sink the message. MrdRsp returns the credit.
- Credit for coherent writes (RbrReq + DtwReq). This credit pool is managed by DCE and DMI has enough buffer space to sink the messages.
- Credit for non-coherent DtwReq. This credit pool is managed by DII/DMI, and the credit is sent back to the AIU using StrReq.

Rbr and Mrd credit mechanism do not need to be updated because the DCE and the DMI are always local to each other. The credit mechanism for non-coherent write cannot be easily modified because StrReq also indicates an ordering point.

3.5.2 CmdReq credit implementation for D2D

3.5.2.1 Local CmdReq credit (i.e., AIU and DCE/DMI/DII in the same chiplet)

There is no change to the handling of local credits.

3.5.2.2 Remote CmdReq credit (i.e., AIU and DCE/DMI/DII in the different chiplets)

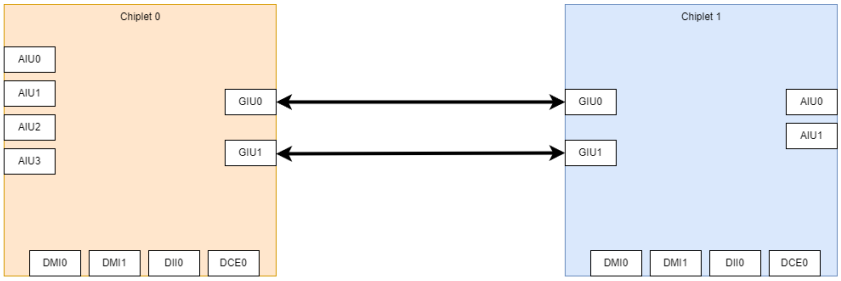
Each AIU will have a register to program the number of credits to each remote chiplet. When programmed, this number of credit will take a buffer space in every DCE/DMI/DII of the remote chiplet without further granularity.

The register will be called Remote Credit Control (RCCR). There will be one per remote chiplet (RCCR_i) and the number is fixed to the number of die in the configuration. It will have a similar structure to other units:

Bits	Field Name	SW Access	HW Access	Descriptions
7:0	CreditLimit	RW	RO	Supports of up to 127 credits. This field needs to be programmed with the number of credits assigned to the remote chiplet this register corresponds to.
9:8	Reserved	RO	RO	Reserved for future extension.
12:10	CounterState	RO	RW	Credit Counter State: 000 (Normal Operation), 001 (Empty), 010 (Negative), 100 (Full), 111 (No connection)
31:13	Reserved	RO	RO	Reserved

TABLE 3-1 XAIURCCR : REMOTE CREDIT CONTROL REGISTER IN THE AIU. ADDRESS OFFSET: 0XC90 + 0X4*I, $i \in [\![0, nDies - 1]\!]$

3.5.2.3 Example of register programming



AIU	Register offset	Register name	Bits	Fields Name	Value
0	0xc90	AIURCCR_0	7:0	CreditLimit	00
	0xc94	AIURCCR_1	7:0	CreditLimit	20
	0xc00	AIUCCR_0	4:0	DCECreditLimit	8
			12:8	DMICreditLimit	8
			20:16	DIICreditLimit	8
	0xc04	AIUCCR_1	4:0	DCECreditLimit	0 (RO)
			12:8	DMICreditLimit	9
			20:16	DIICreditLimit	0 (RO)
1	0xc90	AIURCCR_0	7:0	CreditLimit	00
	0xc94	AIURCCR_1	7:0	CreditLimit	40
	0xc00	AIUCCR_0	4:0	DCECreditLimit	8
			12:8	DMICreditLimit	8
			20:16	DIICreditLimit	8
	0xc04	AIUCCR_1	4:0	DCECreditLimit	0 (RO)
			12:8	DMICreditLimit	10
			20:16	DIICreditLimit	0 (RO)
2	0xc90	AIURCCR_0	7:0	CreditLimit	00
	0xc94	AIURCCR_1	7:0	CreditLimit	60
	0xc00	AIUCCR_0	4:0	DCECreditLimit	8
			12:8	DMICreditLimit	8
			20:16	DIICreditLimit	8
	0xc04	AIUCCR_1	4:0	DCECreditLimit	0 (RO)
			12:8	DMICreditLimit	11
			20:16	DIICreditLimit	0 (RO)
3	0xc90	AIURCCR_0	7:0	CreditLimit	00
	0xc94	AIURCCR_1	7:0	CreditLimit	80
	0xc00	AIUCCR_0	4:0	DCECreditLimit	8
			12:8	DMICreditLimit	8
			20:16	DIICreditLimit	8
	0xc04	AIUCCR_1	4:0	DCECreditLimit	0 (RO)
			12:8	DMICreditLimit	12
			20:16	DIICreditLimit	0 (RO)

AIU	Register offset	Register name	Bits	Fields Name	Value
0	0xc90	AIURCCR_0	7:0	CreditLimit	10
	0xc94	AIURCCR_1	7:0	CreditLimit	00
	0xc00	AIUCCR_0	4:0	DCECreditLimit	6
			12:8	DMICreditLimit	8
			20:16	DIICreditLimit	8
	0xc04	AIUCCR_1	4:0	DCECreditLimit	0 (RO)
			12:8	DMICreditLimit	13
			20:16	DIICreditLimit	0 (RO)
1	0xc90	AIURCCR_0	7:0	CreditLimit	30
	0xc94	AIURCCR_1	7:0	CreditLimit	00
	0xc00	AIUCCR_0	4:0	DCECreditLimit	7
			12:8	DMICreditLimit	8
			20:16	DIICreditLimit	8
	0xc04	AIUCCR_1	4:0	DCECreditLimit	0 (RO)
			12:8	DMICreditLimit	14
			20:16	DIICreditLimit	0 (RO)

Note:
1. AIUCCR_* are existing registers in Ncore 3.7.x
2. AIURCCR_* are new registers to support multi-die architecture from Ncore 3.8.0

Unit	Number of entries reserved for remote traffic	Number of entries used by local traffic
DMI0	AIURCCR_0[chiplet1,AIU0]["credit_limit"] + AIURCCR_0[chiplet1,AIU1]["credit_limit"] = 10+30 =40	AIUCCR_0[chiplet0,AIU0]["DMI0CreditLimit"] + AIUCCR_0[chiplet0,AIU1]["DMI0CreditLimit"] + AIUCCR_0[chiplet0,AIU2]["DMI0CreditLimit"] + AIUCCR_0[chiplet0,AIU3]["DMI0CreditLimit"] = 8+8+8+8 = 32
DMI1		AIUCCR_1[chiplet0,AIU0]["DMI1CreditLimit"] + AIUCCR_1[chiplet0,AIU1]["DMI1CreditLimit"] + AIUCCR_1[chiplet0,AIU2]["DMI1CreditLimit"] + AIUCCR_1[chiplet0,AIU3]["DMI1CreditLimit"] = 9+10+11+12 = 42
DII0		AIUCCR_0[chiplet0,AIU0]["DII0CreditLimit"] + AIUCCR_0[chiplet0,AIU1]["DII0CreditLimit"] + AIUCCR_0[chiplet0,AIU2]["DII0CreditLimit"] + AIUCCR_0[chiplet0,AIU3]["DII0CreditLimit"] = 8+8+8+8 = 32
DCE0		AIUCCR_0[chiplet0,AIU0]["DCE0CreditLimit"] + AIUCCR_0[chiplet0,AIU1]["DCE0CreditLimit"] + AIUCCR_0[chiplet0,AIU2]["DCE0CreditLimit"] + AIUCCR_0[chiplet0,AIU3]["DCE0CreditLimit"] = 8+8+8+8 = 32

Unit	Number of entries reserved for remote traffic	Number of entries used by local traffic
DMI0	AIURCCR_1[chiplet0,AIU0]["credit_limit"] + AIURCCR_1[chiplet0,AIU1]["credit_limit"] + AIURCCR_1[chiplet0,AIU2]["credit_limit"] + AIURCCR_1[chiplet0,AIU3]["credit_limit"] = 20 + 40 +60 + 80=200	AIUCCR_0[chiplet1,AIU0]["DMI0CreditLimit"] + AIUCCR_0[chiplet1,AIU1]["DMI0CreditLimit"] = 8+8 = 16
DMI1		AIUCCR_1[chiplet1,AIU0]["DMI1CreditLimit"] + AIUCCR_1[chiplet1,AIU1]["DMI1CreditLimit"] = 13+14 = 27
DII0		AIUCCR_0[chiplet1,AIU0]["DII0CreditLimit"] + AIUCCR_0[chiplet1,AIU1]["DII0CreditLimit"] = 8+8= 16
DCE0		AIUCCR_0[chiplet1,AIU0]["DCE0CreditLimit"] + AIUCCR_0[chiplet1,AIU1]["DCE0CreditLimit"] = 6 + 7 = 13

FIGURE 3-1 EXAMPLE OF CREDIT PROGRAMING

3.5.2.4 Quantitative analysis of the number of entries in the skid buffers for CmdReq

Let us assume a system with 4 chiplets, 20 AIU/Chiplets and an average round trip latency of about 60 cycles to remote target.

The buffer space needed in DMI, DII and DCE to accommodate those credit is: 20 x 3 x 60 = 3600 entries.

Each entry is about 25B which leads to a total memory size of 90kB.

Those entries would be on top of what is required by the local traffic and would constitute an already large system.

Assuming an absolute worst-case scenario (not realistic) of 32 AIU/chiplet and a round-trip latency of 128 cycles to remote targets we obtain: 32 x 3 x 128 = 12,288 entries. (300kB)

The Skid buffer implementation target is a two ports SRAM fifo. In most cases it should be expected that this fifo will be bypassed.

3.5.3 SnpReq credit implementation for D2D

3.5.3.1 Local SnpReq (i.e., DCE and AIU are in the same chiplet)

There is no change to the handling of local credits.

3.5.3.2 Remote SnpReq (i.e., DCE and AIU are in different chiplet)

It is proposed that the credit is assigned for the entire duration of the snoop. Consequently, the GIU will not need to return credits or need skid buffers. Instead, each DCE will track the remote credit on a per chiplet granularity. This number of credits is fixed for all the assemblies and does not depend on the chiplet. We will call those credit RemoteSnpCredit.

Each DCE will be able to send a maximum of RemoteSnpCredit to each chiplet and consequently each AIU will need to have nDCE x RemoteSnpCredit buffer space to sink those messages, in which nDCE is the total number of DCE on the entire configuration.

The target implementation is a large two ports SRAM fifo.

This means that the DCE will need Credit Counters for the remote chiplet. Maestro provides to the DCE with the number of Chiplets.

3.5.3.3 Quantitative analysis of the number entries in the skid buffers for SnpReq.

Let’s assume a system with 4 chiplets and 16 DCE per Chiplet. Let’s assume a round-trip latency of 60 cycles for remote SnpReq.

The buffer space to accommodate those credits is: 3 x 16 x 60 = 2,800 entries.

Each message is roughly 25B for a total of 72 kB.

Increasing the round-trip latency to 128 like for commands gives 6144 entries (153 kB).

3.6 Ordering requirement

GIU is not playing any protocol role anymore and will process every transaction in the order it is received.

3.7 AIU/Snoop filter assignment

3.7.1 Introduction

The support of multiple assembly is not planned for Ncore 3.8 but in order to prepare for that support the assignment between AIUs and snoop is not resolved at design time and passed as a parameter/tie-offs to DCE. Instead, it will be controlled by a set of registers which allow the mapping of a CachingAgentId and an FUnitId.

This scheme will also be used in single die system but the reset value of the registers will be prepopulated by values from the snoop filter object.

3.7.2 CachingAgentId

The CachingAgentId refers to a location in the directory. It is indexed by incrementing snoop filter index and incrementing location within the snoop filter.

For example, assume a configuration with N snoop filters. Each snoop filter S_i contains K_i caching agent where i ranges from 0 to N-1.

The caching agent in Snoop filter s, index c will be assigned the following CachingAgentId:

$$CachingAgentId = \sum_{i=0}^{s-1} K_i + c$$

3.7.3 CachingAgentId to AIU mapping. DCE CachingAgentId to AIU Mapping register

A new set of registers is introduced to allow the mapping of a CachingAgentId to GlobalId of an AIU.

Bits	Field Name	SW access	HW access	Reset value	Descriptions
9:0	GlobalID of caching agent 2*i	RW	RO	0x0	If assigned, bit[9:0] == {ChipletId, FUnitId} of the AIU assigned to CachingAgentId 2*i
10	Valid	RW	RO	0x0	1'b1 indicates it is a valid mapping. 1'b0, otherwise
15:11	Reserved	RO	RO	0x0	Reserved
25:16	GlobalID of caching agent 2*i+1	RW	RO	0x0	If assigned, bit[9:0] == {ChipletId, FUnitId} of the AIU assigned to CachingAgentId 2*i+1
26	Valid	RW	RO	0x0	1'b1 indicates it is a valid mapping. 1'b0, otherwise
31:26	Reserved	RO	RO	0x0	Reserved

TABLE 3-2 DCEUCAMR[I] DCE CACHINGAGENTID TO AIU MAPPING REGISTER. ADDRESS OFFSET: 0X800+I*0X4, $i \in [\![0,63]\!]$

Those registers allow software to assign each coherent agent to a CachingAgentId.

NOTE:
CachingAgentId corresponds to the bit position of the DCE Directory Manager's Sharer Vector which is a concatenation of each of its Snoop Filter's Sharer Vector.
For example:
DCE Directory Manager has 2 snoop filters: SF0 has 2 sharers, SF1 has 3 sharers.
Directory Manager's Sharer Vector [4:0] = {SF1's Sharer Vector [2:0], SF0's Sharer Vector [1:0]}
CachingAgentId=n corresponds to Directory Manager's Sharer Vector[n] where n is in the range of 0 to 4.

3.7.4 DCEUSER DCE Snoop enable register update

The snoop enable register is now indexed using the CachingAgentId instead of the NUnitId. Bit [i] of DCEUSER[j] maps to $CacheId = 32.j + i$, where $i \in [\![0,31]\!], j \in [\![0,3]\!]$.

The maximum number of agents tracked is 128 (32/chiplet).

3.8 Exclusive monitor assignment.

3.8.1 Coherent exclusive monitors

The basic monitor will now use a concatenation of the CachingAgentId introduced in the previous paragraph and of the LPID (/AXI Id) carried by the mpf2 field of the CmdReq to index into the monitor vector.

Consequently, Maestro will need to provide the largest number of processors of any agent in the entire systeNon-coherent exclusive monitors

Non-coherent exclusive monitor will need to track which ChipletId the transaction are coming from instead of only he FUnitId.

3.8.2 Non coherent exclusive monitors

Non coherent exclusive monitors will use the GlobalFunitId in order to match an agent.

3.9 DVM handling

3.9.1 Introduction and motivation

In Ncore 3.7 all DVM CmdReq request go through a single unit (DVE) which is responsible for the ordering and unicast of the snoop DVM.

Ncore 3.8 introduces the concept of multiple DVM ordering points. The idea is that an AIU always sends its CmdReq to its local DVE. DVE is then responsible for unicasting the SnpDvm to the local AIU targets (like before) and to remote DVEs. The remote DVE are responsible for the unicast to the AIU local to them, aggregate their responses and respond to the initiating DVE with a single response. In the rest of the document, we will call this approach “Distributed DVM”.

The introduction of this new concept generates the possibility for two agents in different systems to receive the snoop DVM in a different order. Although, in ARM AMBA CHI specification explicitly mentions that an MN must guarantee that all agents see the same order for the same cache line, it has been confirmed by ARM that having multiple serialization point should not create an issue (see appendix).

In order to mitigate the risk related to the distribution of DVM messages to multiple DVE, the system will also allow operating in a mode where all DVM CmdReq will target the same DVE while the unicast of the SnpReq will still be the responsibility of the local DVEs. In the rest of the document, we will call this approach “Centralized DVM”.

The switch from one mode to the other will only be allowed at boot time (before any DVM traffic has occurred) by writing the registers described in paragraph 3.9.3, (Table 3-4 DVEURDVEFUIDR Remote DVE FUnitId register. Address OFFSET: 0xc.) and the default behavior will be distributed.

FIGURE 3-2 shows a high-level view of the difference between Centralized and Distributed approaches. In the centralized approach every AIU in the system sends the DVM CmdReq (black arrows) to the DVE of chiplet 1 while in the Distributed approach it is sent to the local DVE. The functionality of the DVE should be identical in both approaches.

It is recommended but not required that logic and the register related to multi die/chiplet systems are only present in hardware in such configuration.

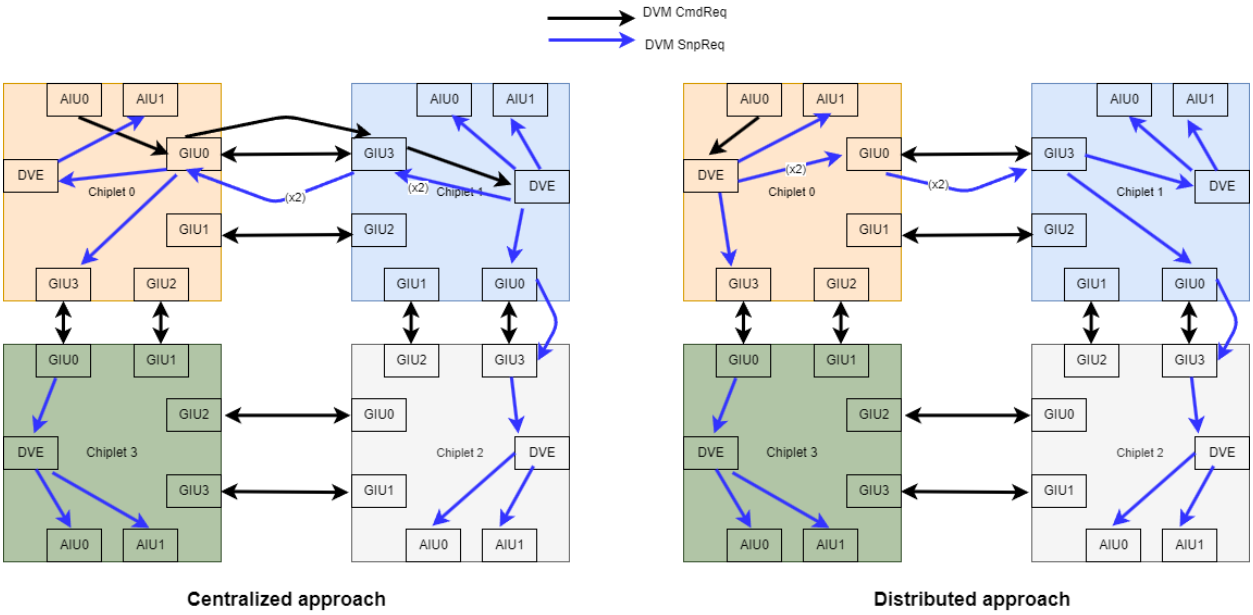


FIGURE 3-2 HIGH LEVEL VIEW COMPARISON BETWEEN CENTRALIZED AND DISTRIBUTED APPROACHES

3.9.2 DVE responsibilities for handling DVM requests.

The responsibilities of DVE for DVM handling can be summarized as follows:

- Receive and process the CmdReq that arrives at its port.
- Unicast the SnpReq to the local target for which the Snoop enable register bit is set to 1 when receiving a CmdReq or a SnpReq.
- Unicast the SnpReq to the remote DVE in chiplet for which the register bit described in section 3.9.6, Table 3-6 DVEUDRSER DVE Remote DVM Snoop Enable Register. Address offset: 0x14 is set to 1 when receiving a CmdReq.
- Aggregate all the SnpRsp from the local target.
- Issue a single SnpRsp back to the DVE if the sequence was initiated in a remote die.
- Aggregate all the SnpRsp from the local targets and remote DVE when the SnpReq was initiated by a CmdReq and issue the response back to the initiating AIU (CmpRsp).

Those responsibilities do not depend on the choice of a centralized or a distributed approach.

3.9.3 Home DVE register.

A new register is introduced in every AIU (regardless of their ability to process DVM). It contains the Global Id (concatenation of ChipletId and FUnitId) of the DVE that will receive the DVM CmdReq.

In order to operate in Centralized mode, all the registers in the system will contain the same Global Id regardless on which chiplet they are in.

In order to operate in Distributed mode, it will contain the Global Id of the local DVE.

Bits	Field Name	SW Access	HW Access	Reset Value	Descriptions
10:0	HomeDVEGlobalId	RW	RO	Local DVE global Id	Contains the Global Id ({ChipletId, FUnitId}) of the DVE which will process the DVM commands coming from this AIU. To operate in distributed mode, the DVE from the current chiplet must be used. To operate in Centralized mode the same DVE must be used for every AIU across all the chiplets. By default, the system will operate in distributed mode. If there is only one chiplet do not modify this register.
12:11	HomeDVELinkId	RW	RO	x0	This field is used to set which LinkId should be used to route the CmdReq to the remote DVE when The HomeDVEGlobalId is programed to be remote. The default behavior is to use LinkId = 0
31:13	Reserved	RO	RO	x0	Reserved

TABLE 3-3 XAIUHOMEDVE REGISTER PRESENT IN EVERY AIU. ADDRESS OFFSET: 0X8

3.9.4 Remote DVE FUnitId register

DVE will need to be aware of the remote DVE in the system in order to unicast the SnpReq to them. Consequently, a new register is introduced in DVE for software to program their FUnitId into it.

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
7:0	Chiplet0DVEFUnitId	RW	RO	Chiplet0 DVE FUnitId	8-bit FUnitId for the DVE located in chiplet 0
15:8	Chiplet1DVEFUnitId	RW	RO	Chiplet1 DVE FUnitId	8-bit FUnitId for the DVE located in chiplet 1
23:16	Chiplet2DVEFUnitId	RW	RO	Chiplet2 DVE FUnitId	8-bit FUnitId for the DVE located in chiplet 2
31:24	Chiplet3DVEFUnitId	RW	RO	Chiplet3 DVE FUnitId	8-bit FUnitId for the DVE located in chiplet 3

TABLE 3-4 DVEURDVEFUIDR REMOTE DVE FUNITID REGISTER. ADDRESS OFFSET: 0XC.

3.9.5 DVE DVM Domain Configuration register

885

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
0	Chiplet0IsInDVMDomain	RO	RO	0x1 if Chiplet 0 supports DVM 0x0 otherwise.	This register is set to 1 if Chiplet 0 was defined at design time and it supports DVM
1	Chiplet1IsInDVMDomain	RO	RO	0x1 if Chiplet 1 supports DVM 0x0 otherwise.	This register is set to 1 if Chiplet 1 was defined at design time and it supports DVM
2	Chiplet2IsInDVMDomain	RO	RO	0x1 if Chiplet 2 supports DVM 0x0 otherwise.	This register is set to 1 if Chiplet 2 was defined at design time and it supports DVM
3	Chiplet3IsInDVMDomain	RO	RO	0x1 if Chiplet 3 supports DVM 0x0 otherwise.	This register is set to 1 if Chiplet 3 was defined at design time and it supports DVM
28:4	Reserved	RO	RO	x0	Reserved
31:29	DefaultConfigNumber	RO	RO	x0	This config number is provided by Maestro and refers to a collateral generated to help with programming of the Ncore registers.

TABLE 3-5 DVEUDVMDCR DVM DOMAIN CONFIGURATION REGISTER IN DVE. ADDRESS OFFSET: 0X10

This register in DVE allows software to discover which chiplets can receive DVM snoops and are part of the same DVM domain. For Ncore 3.8.0 release, the default configuration will be such that there is one and only one domain, this register is meant to evolve to help software find which chiplet are in the same DVM domain.

3.9.6 Remote DVM Snoop Enable Register and Domain Configuration Register

Each DVE needs a way to decide if a SnpReq needs to be sent to a remote DVE. This register should be set up so that if a chiplet is absent, powered down or does not have DVM support it does not receive DVM SnpReq from other chiplet. It is programmed at boot and updating it after boot will require careful software controls of all the enable registers in the system.

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
0	Chiplet0DVMEn	RW	RO	x0	1 bit which enables the current DVE to send DVM snoops to the agent in chiplet 0 through its DVE. If the current Chiplet is 0 it has no effect.
1	Chiplet1DVMEn	RW	RO	x0	1 bit which enables the current DVE to send DVM snoops to the agent in chiplet 1 through its DVE. If the current Chiplet is 1 it has no effect.
2	Chiplet2DVMEn	RW	RO	x0	1 bit which enables the current DVE to send DVM snoops to the agent in chiplet 2 through its DVE. If the current Chiplet is 2 it has no effect.
3	Chiplet3DVMEn	RW	RO	x0	1 bit which enables the current DVE to send DVM snoops to the agent in chiplet 3 through its DVE. If the current Chiplet is 3 it has no effect.
28:4	Reserved	RO	RO	x0	Reserved
31:29	CurrentConfigNumber	RW	RO	x0	This config number is provided by Maestro and refers to a collateral generated to help with programming of the Ncore registers. It must be populated by software to help with debuging. It is not used internally by Ncore.

TABLE 3-6 DVEUDRSER DVE REMOTE DVM SNOOP ENABLE REGISTER. ADDRESS OFFSET: 0X14

Software must use the help of DVMDCR (Table 3-5 DVEUDVMDCR DVM Domain configuration register in DVE. Address offset: 0x10) in order to program DVERSER. In particular, it is not allowed to program a chiplet as enabled in DVERSER if it shows in DVMDCR that it is not part of the DVM domain.

Depending on the subsystem used, Maestro will provide various configuration files which correspond to a Configuration Number to help with programming this register. It will also assume all the chiplets are in the same DVM domain.

However, it is allowed for a customer to program this register as to have multiple DVM domain. For example, it can keep remote DVM always disable and its chiplet will be independent from a DVM standpoint.

3.9.7 DVE DVM LinkId register

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
1:0	RemoteLinkIdChiplet0	RW	RO	0x0	This register is used to populate the LinkId field when DVE sends DVM snoops to a remote target in chiplet 0. LinkId indexes links connecting the same Chiplet.
3:2	RemoteLinkIdChiplet1	RW	RO	0x0	This register is used to populate the LinkId field when DVE sends DVM snoops to a remote target in chiplet 1. LinkId indexes links connecting the same Chiplet.
5:4	RemoteLinkIdChiplet2	RW	RO	0x0	This register is used to populate the LinkId field when DVE sends DVM snoops to a remote target in chiplet 2. LinkId indexes links connecting the same Chiplet.
7:6	RemoteLinkIdChiplet3	RW	RO	0x0	This register is used to populate the LinkId field when DVE sends DVM snoops to a remote target in chiplet 3. LinkId indexes links connecting the same Chiplet.
31:2	Reserved	RO	RO	0x0	Reserved

TABLE 3-7 DVEUDVMLIR DVE DVM LINKID REGISTER. ADDRESS OFFSET: 0X18

This register is used by DVE to populate the LinkId field of the DVM snoop request sent to remote DVEs.

3.9.8 Semantic, credit and transaction diagram updates

3.9.8.1 Update to the mpf2 field semantic of the SnpReq.

The mpf2 field of the SnpReq will always carry the GlobalId of the AIU which initiated the CmdReq. As we will see in the transaction diagram below it will allow DVE to avoid snooping that AIU when it receives a SnpReq from a remote DVE in the Centralized approach.

3.9.8.2 Update to the connectivity

Because the DVE becomes responsible for unicasting of the SnpReq received from a remote DVE to the local AIUs, it is required for DVE to be able to receive SnpReq and send SnpRsp which was not the case in Ncore 3.7.

This connectivity change must only apply to multi-die systems.

3.9.8.3 Impact on the number of CmdReq skid buffers in DVE

The CmdReq skid buffer in the DVE will need to be sized for the worst-case scenario i.e., for the centralized approach. Consequently, the number of skid buffer entries in all DVEs in the system needs to be the sum of all the DVM credit across all the AIUs in the entire system.

3.9.8.4 Impact on the number of SnpReq skid buffers in the AIUs

Similarly, as for the CmdReq, the worst-case scenario needs to be accounted for. Consequently, the number of snoop skid buffers will be the sum over all the chiplet of the number of STT entries in DVE.

3.9.8.5 2 Snoops DVM request for centralized approach

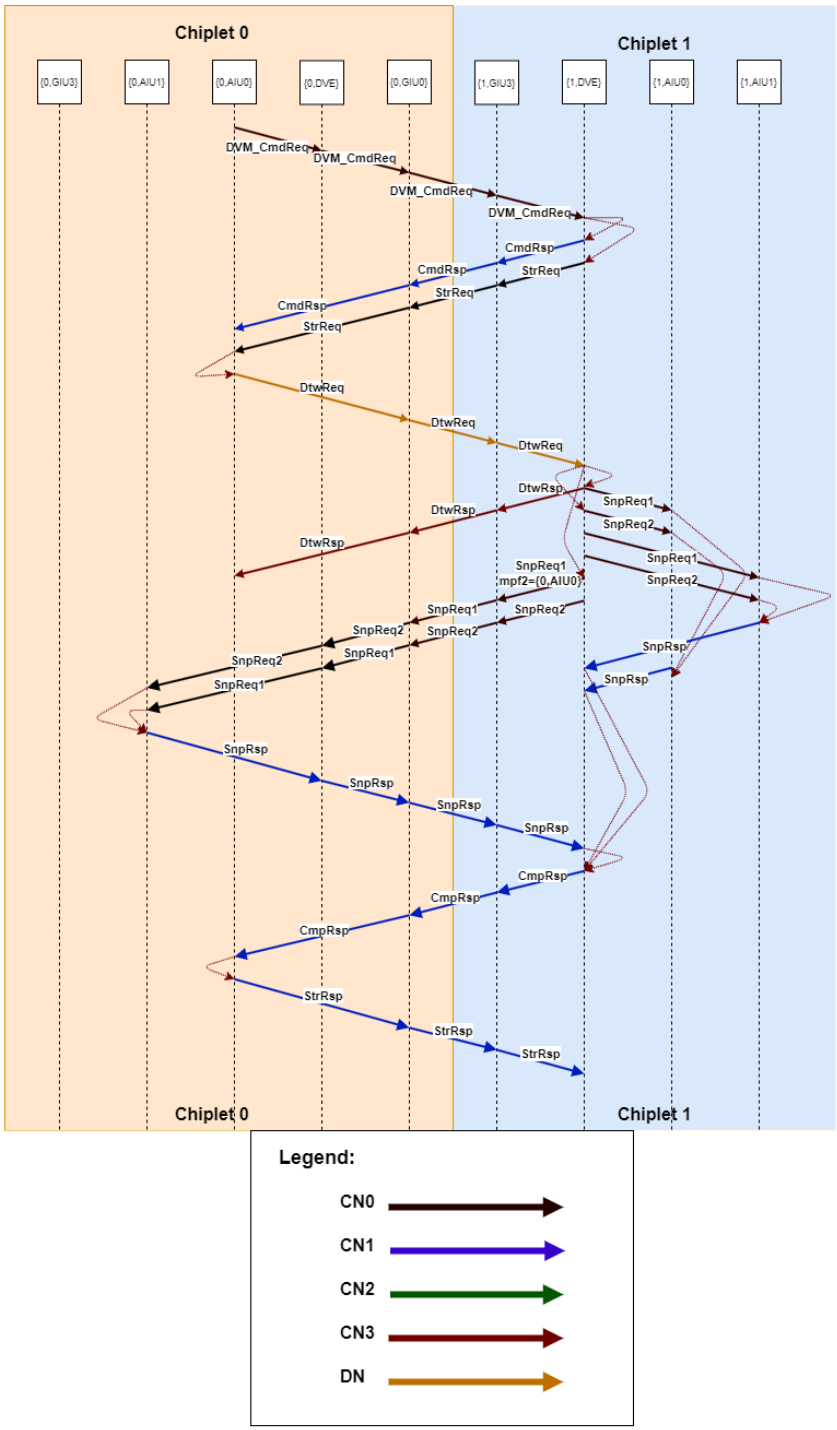
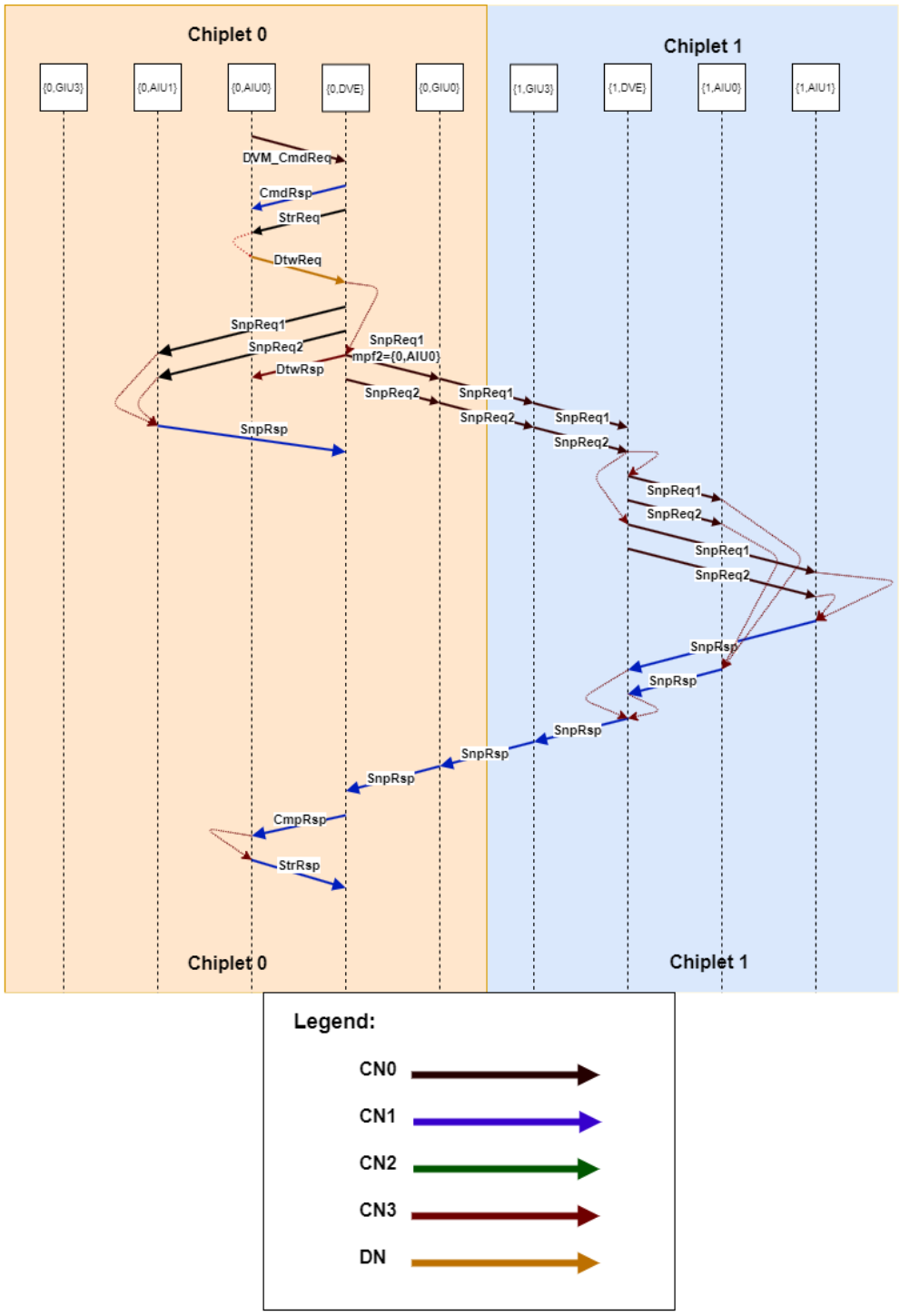


FIGURE 3-3 EXAMPLE OF A DVM CMDREQ WITH THE CENTRALIZED APPROACH FOR TWO CHIPLETS.

The transaction diagram shows an example of a DVM flow in the case of two chiplets. The DVE of chiplet 1 has been programmed to receive the DVM CmdReq of the entire system. Consequently, AIU0 of chiplet 0 sends its command to it. The DVE of chiplet 1 unicasts SnpReq to its local targets per the value of the Snoop enable register and to the remote DVE per Table 3-6 DVEUDRSER DVE Remote DVM Snoop Enable Register. Address offset: 0x14.

The DVE of chiplet 0 then unicast the SnpReq to the local targets per its own snoop enable register. Note that without the introduction of the mpf2 field the DVE of chiplet 0 would not be aware that the initiator was the AIU0 of Chiplet 0 and it would send a SnpReq to it as well. This is the reason why the mpf2 field is updated to contain the GlobalId of the initiator of the CmdReq.

3.9.8.6 2 Snoops DVM request for distributed approach



951 FIGURE 3-4 EXAMPLE OF DVM REQUEST FLOW IN DISTRIBUTED MODE FOR A SYSTEM WITH 2 CHIPLETS.

Figure 3-4 show an example of DVM flow when the system operates in distributed mode i.e., the AIUs always send the DVM CmdReq to the local DVE.

3.10 Domain attachment

3.10.1 Introduction

For Ncore 3.8 there is only one coherency domain, consequently an AIU when triggering an attachment (either using the software-driven CSR trigger mechanism or using the native interface's SYSCO trigger mechanism) must register to all DCE and DVE in the system. In order to limit the number of units which need to be aware of the system it was decided that DVE will be responsible for forwarding attachment/de-attachment request (using the SysReq semantic) to the DCE in the same chiplet and to the DVE in the remote chiplet.

For DVM purposes, DVE will continue to update its Snoop enable register for the local DVM target that it takes care of.

Ncore 3.8 will continue to support a purely software driven attachment by writing directly to the registers in DCE.

3.10.2 AIU responsibilities

There is a change in the responsibility of the AIUs. Instead of unicasting to every DCE and to DVE the SysReq attach message, they will only send the request to the local DVE. This change will apply to all systems (multi-die and single die systems).

Note that it is always sent to the local DVE irrespective of the choice made for DVM handling (distributed/centralized).

For an attachment to be successful DCE snoop filter must have been programmed first.

3.10.3 DVE responsibilities

A DVE will have the following responsibilitie with regards to attachment/de-attachment of agent using the SysReq handshake:

- Unicast to other DVE in the system if the Initiator of the SysReq is a local AIU per Table 3-9 DVEUCAER DVE coherent Attach Enable Register. Address Offset: 0x24. and the AIU is a caching agent.
- Unicast to local DCE the attachment/de-attachment requests if the initiator is a remote DVE or if it is a local caching agent.
- Aggregate all the SysRsp and respond back to the initiator of the SysReq with a single message.
- Update its Snoop Enable register for DVM capable agents.

- Propagate cm_status errors all the way back to the AIU.

3.10.4 List of local caching agent LocalCachingAgent

A list of the FUnitId of all the local caching agent will be provided to DVE by Maestro. The parameter is called LocalCachingAgent and is an array which contains the FUnitId of all the local caching agents.

Upon receiving an attach message DVE will check the InitiatorId of the SysReq.attach. If the chiplet field is decoded as being local and the FUnitId is included in the list of local caching agents, DVE will unicast the attach to every local DCE and to remote DVEs.

The intent for introducing this list of caching agent in DVE is to avoid sending unnecessary SysReq to DCE and remote DVE and to keep a clear error condition for when a SysReq.attach is sent to DCE but does not match any known agent. Without this list, DCE would not be able to know if it received an attach from an unknown agent because snoop filter assignment register was not programed correctly (software error) or because it simply is not a caching agent.

3.10.5 List of local DVM agent LocalDVMAgentFUnitId and LocalDVMAgentNUnitId

A list of the FUnitId of all the DVM agent will be provided to DVE by Maestro. The parameter is called LocalDVMAgentFUnitId (respectively LocalDVMAgentNUnitId) and is an array which contains the FUnitId (respectively NUnitId) of all the local DVM agents.

Upon receiving an attach message DVE will check the InitiatorId of the SysReq.attach. If the chiplet field is decoded as being local and the FUnitId is included in the list of local DVM agents, DVE will update its snoop enable register for that agent according to its NUnitId.

This parameter currently is a tie-off to DVE (uSysId_f_unit_id and uSysNodeId_n_unit_id) which will be removed because it does not need to be a tie-off.

3.10.6 DVE Coherent Domain Configuration Register

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
0	Chiplet0CohDom	RO	RO	x1 if chiplet 0 exists	This register is set to 1 if Chiplet 0 was defined at design time.

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
1	Chiplet1CohDom	RO	RO	x1 if chiplet 1 exists	This register is set to 1 if Chiplet 1 was defined at design time.
2	Chiplet2CohDom	RO	RO	1 if chiplet 2 exists	This register is set to 1 if Chiplet 2 was defined at design time.
3	Chiplet3CohDom	RO	RO	1 if chiplet 3 exists	This register is set to 1 if Chiplet 3 was defined at design time.
28:4	Reserved	RO	RO	x0	Reserved
31:29	DefaultConfigNumber	RO	RO	x0	This register contains the default configuration number which corresponds to a register configuration file provided at design time by maestro.

1014 TABLE 3-8 DVEUCDCR DVE COHERENT DOMAIN CONFIGURATION REGISTER. ADDRESS OFFSET: 0X20

1017 ## 3.10.7 DVE Coherent Attach Enable register

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
0	Chiplet0CohEn	RW	RO	x0	1 bit which enables the current DVE to forward Coherent attachment/de-attachment requests to the DVE of chiplet 0.
1	Chiplet1CohEn	RW	RO	x0	1 bit which enables the current DVE to forward Coherent attachment/de-attachment requests to the DVE of chiplet 1.
2	Chiplet2CohEn	RW	RO	x0	1 bit which enables the current DVE to forward Coherent attachment/de-attachment requests to the DVE of chiplet 2.
3	Chiplet3CohEn	RW	RO	x0	1 bit which enables the current DVE to forward Coherent attachment/de-attachment requests to the DVE of chiplet 3.
28:4	Reserved	RO	RO	x0	Reserved

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
31:29	CurrentConfigNumber	RW	RO	x0	This register contains the current configuration number which corresponds to a register configuration file provided at design time by Maestro.

TABLE 3-9 DVEUCAER DVE COHERENT ATTACH ENABLE REGISTER. ADDRESS OFFSET: 0X24.

This register is used to control the forwarding of Attachment/De-attachment requests from a local DVE to the remote DVE of other chiplets.

3.10.8 DVE System Coherency Attachment LinkId Register

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
1:0	RemoteSysCoAttachLinkIdChiplet0	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.attach to remote chiplet 0. LinkId indexes links connecting the same Chiplet.
3:2	RemoteSysCoAttachLinkIdChiplet1	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.attach to remote chiplet 1. LinkId indexes links connecting the same Chiplet.
5:4	RemoteSysCoAttachLinkIdChiplet2	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.attach to remote chiplet 2. LinkId indexes links connecting the same Chiplet.
7:6	RemoteSysCoAttachLinkIdChiplet3	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.attach to remote chiplet 3. LinkId indexes links connecting the same Chiplet.
31:2	Reserved	RO	RO	0x0	Reserved

TABLE 3-10 DVEUSCALR DVE SYSTEM LINKID REGISTER. ADDRESS OFFSET: 0X2C.

This register is used to populate the LinkId field of the SysReq.attach/SysReq.detach sent by a DVE to a remote DVE. SysRsp will use the same LinkId as the SysReq which created it.

There is no reason to modify this value in general and LinkId 0 will always work unless the corresponding GIU was powered down or the physical link had an error.

3.10.9 RequesterGlobalId: Field addition to SysReq

The RequesterGlobalId field is added to SysReq which carries the GlobalId of the AIU which initiated the SysReq. It allows the remote DCE to know which AIU is trying to attach/de-attach. This information used to be conveyed by the InitiatorId of the SysReq but was not available anymore because DVE is now responsible for the unicast to all DCE.

3.10.10 DCE responsibilities

DCE's responsibility is to update its snoop enable register when it receives a SysReq attach/de-attach. To do so it matches the GlobalId carried by the RequesterGlobalId field against the content of DCEUCAM (Table 3-2 DCEUCAMR[i] DCE CachingAgentId to AIU mapping Register. Address offset: 0x800+i*0x4, $i \in [\![0,63]\!]$). This will allow it to obtain the CachingAgentId per the mapping described in 3.7.3. DCE will then write the bit in the Snoop enable register which corresponds to the CachingAgentId which matched the RequesterGlobalId per the mapping described in 3.7.4.

3.10.11 High level view of the attachment process in a multi-die system

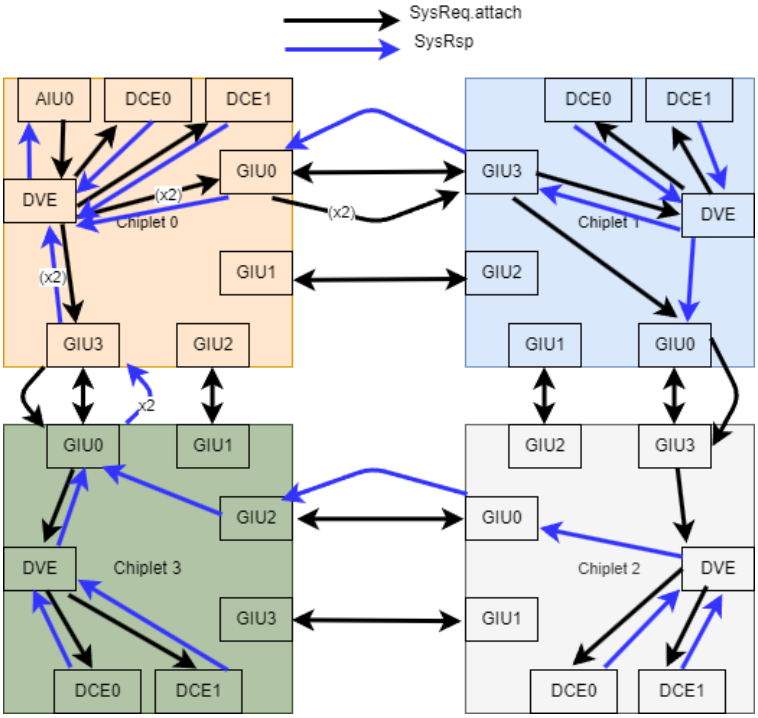


FIGURE 3-5 HIGH LEVEL VIEW OF AN ATTACHMENT SEQUENCE IN A 4 DIE SYSTEM

In this example the AIU0 of chiplet 0 starts an attachment request and sends it to its local DVE. This DVE then forwards this attachment request to every local DCE and to every other DVE in the system for which the register DVEUCAER was set to 1. Those DVE then unicast the SysReq to every local DCE, collect their responses and send a single response back to the DVE of chiplet 0.

For coherency domain purposes it is not allowed to have multiple domains and the register of Table 3-9 DVEUCAER DVE coherent Attach Enable Register. Address Offset: 0x24. must include every chiplet in the system.

3.11 System event handling

3.11.1 Introduction

DVE handles event distribution to the event receiver of the system in Ncore 3.7. A D2D system will extend its functionality and each DVE in each chiplet will be responsible for the distribution of an event to the remote DVE and to the local event receivers.

3.11.2 DVE responsibility

From the point of view of remote DVE, DVE will now also be an event sender.

The DVE will have the following responsibilities:

- Collect the SysReq.event from the event sender in the system (same as Ncore 3.7)
- Unicast the event to all local event receiver target (same as Ncore 3.7)
- If the SysReq.event was sent from a local event sender, unicast the event to the remote DVE which are enabled (see Remote Event Enable register 3.11.3).
- Collect the SysRsp and issue a single SysRsp back to the event sender. Remote System Event Enable register.

3.11.3 DVE remote event enable register

A new register is introduced to DVE to control the distribution of events to remote DVEs.

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
0	Chiplet0EventEn	RW	RO	x0	1 bit which enables the current DVE to forward event requests to the agent in chiplet 0 through its DVE
1	Chiplet1EventEn	RW	RO	x0	1 bit which enables the current DVE to forward event requests to the agents in chiplet 0 through its DVE
2	Chiplet2EventEn	RW	RO	x0	1 bit which enables the current DVE to forward event requests to the agents in chiplet 0 through its DVE
3	Chiplet3EventEn	RW	RO	x0	1 bit which enables the current DVE to forward event requests to the agents in chiplet 0 through its DVE
31:4	Reserved	RO	RO	x0	Reserved

TABLE 3-11 DVEURSEE DVE REMOTE EVENT ENABLE REGISTER. ADDRESS OFFSET: 0X30.

This register is used by DVE to decide if an event originating from a local target needs to be forwarded to a remote DVE.

3.11.4 DVE Remote System event LinkId register

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
1:0	RemoteSysEvLinkIdChiplet0	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.event to remote chiplet 0. LinkId indexes links connecting the same Chiplet.

Bits	Field Name	SW access	HW Access	Reset value	Descriptions
3:2	RemoteSysEvLinkIdChiplet1	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.event to remote chiplet 1. LinkId indexes links connecting the same Chiplet.
5:4	RemoteSysEvLinkIdChiplet2	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.event to remote chiplet 2. LinkId indexes links connecting the same Chiplet.
7:6	RemoteSysEvLinkIdChiplet3	RW	RO	0x0	This register is used to select the LinkId that DVE must use to send SysReq.event to remote chiplet 3. LinkId indexes links connecting the same Chiplet.
31:2	Reserved	RO	RO	0x0	Reserved

TABLE 3-12 DVEURSELR DVE REMOTE SYSTEM EVENT LINKID REGISTER. ADDRESS OFFSET: 0X34.

3.11.5 Credit requirement

DVE needs to be able to receive events from remote chiplet. Consequently, the size of the SysReq skid buffer will be the local number of event senders plus the number of chiplet minus one.

Each Event receiver will still only see a maximum of one event coming from the local DVE.

3.11.1 High level view of event distribution in a multi-die Ncore

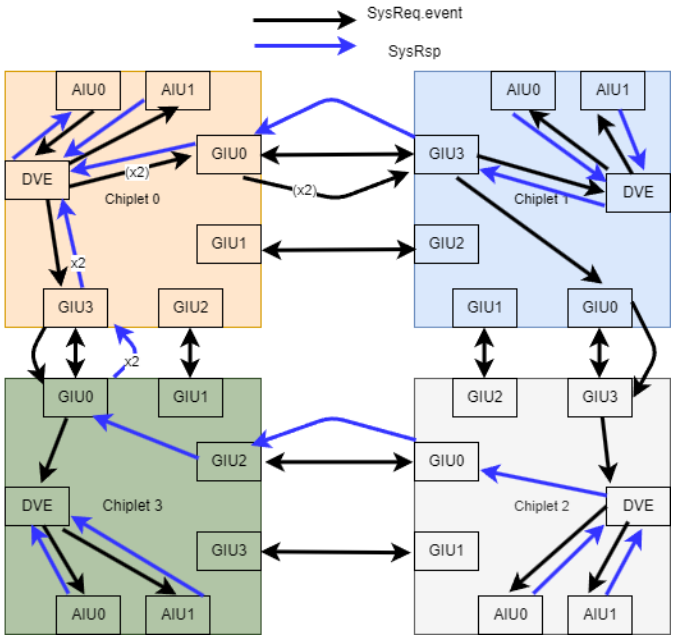


FIGURE 3-6 HIGH LEVEL VIEW OF EVENT DISTRIBUTION IN A 4 DIE SYSTEM

In the example above the register of Table 3-11 DVEURSEE DVE Remote Event Enable register. Address offset: is programmed to distribute the event to every die in the system. The event is initiated by the AIU0 of chiplet 0 and is sent to other chiplets by the DVE of chiplet0. The DVE of chiplet 1,2 and 3 are then responsible to unicast the event to every event receiver in their respective chiplet, to collect their response and to issue a single response back to the DVE of chiplet 0 which responds to AIU0.

3.12 Transaction diagrams

3.12.1 Configuration used for the diagrams

For the purpose of this section the configuration will be a 2x2 mesh. Each chiplet is connected using 2 GIUs part of the same interleaving group.

The InitiatorId respectively TargetId will always be the {FUnitId, PortID} of the unit from which the transaction is sent respectively received. Consequently, they are not mentioned explicitly in most flows.

If a message/field is not annotated, it means that either it is unchanged compared to Ncore 3.7 or that it is unchanged compared to the previous hop.

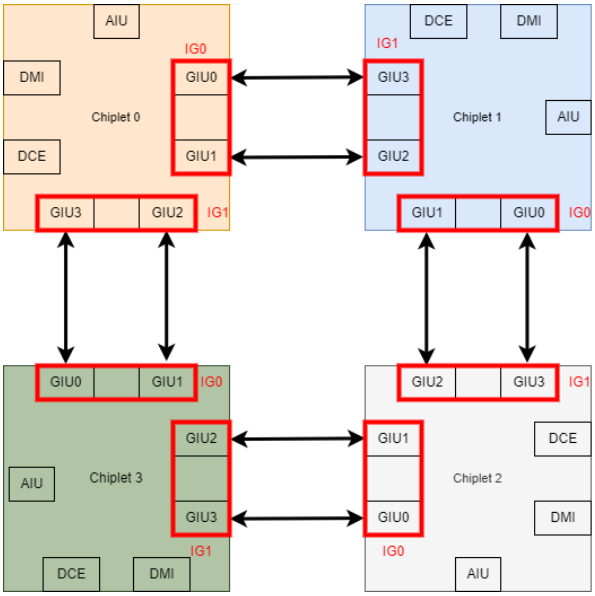


FIGURE 3-7 CONFIGURATION USED FOR THE TRANSACTION DIAGRAMS

3.12.2 Transaction diagram conventions

- All units are represented, specifically GIU on each side of the D2D link are shown as separate units.
- The background color separates the different chiplets, and they match the color used in the configuration.
- The colors of the arrows indicate which network each transaction is in. For GIU to GIU transfer they match the virtual channel.
- The dotted red line indicates dependencies between messages.

It should be noted that the main new feature that affects the protocol diagrams is the hierarchical address map. For that reason, we will illustrate the mechanism for a non-coherent read with route through. This mechanism will be applied to every Ncore flows which are unchanged.

3.12.3 Non-Coherent read flow

Figure 3-8 shows a non-coherent read with route through. It shows how the FUnitId portion of the TargetId is only populated when the CmdReq reaches its destination chiplet and remains untouched otherwise.

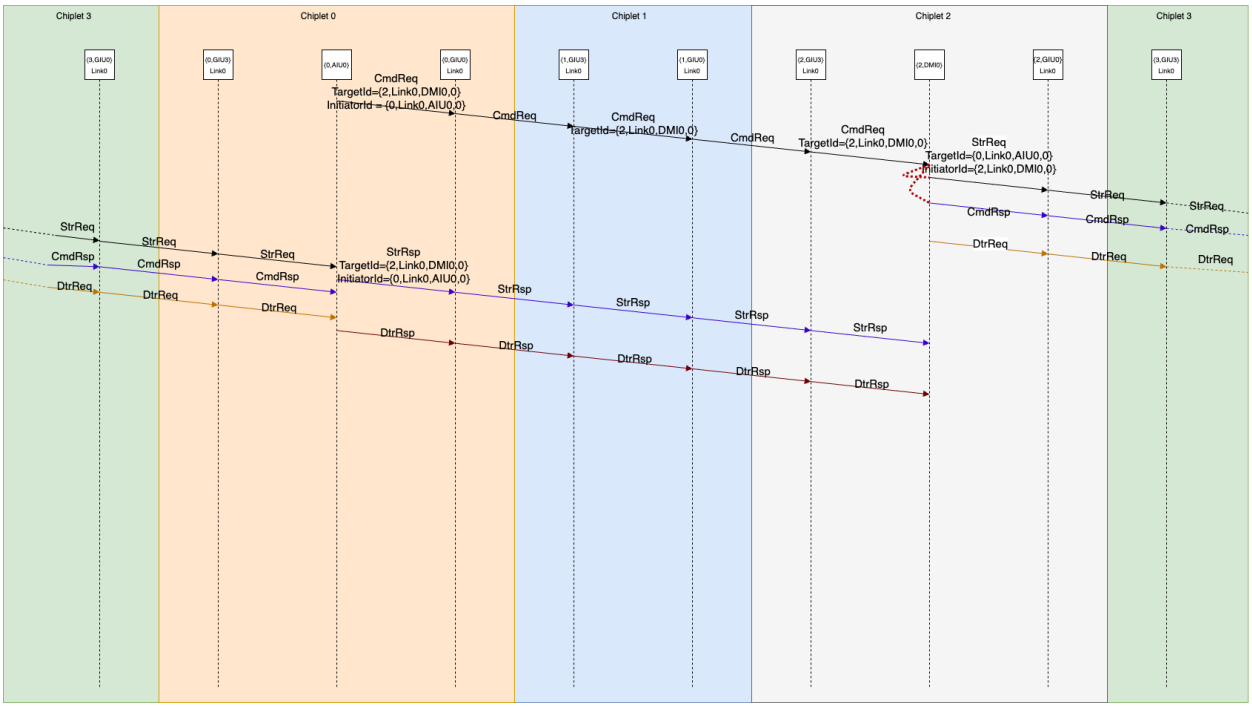


FIGURE 3-8 NON-COHERENT READ WITH ROUTE THROUGH

3.12.4 Coherent read flow

Figure 3-9 shows an example in which the AIU0 from Chiplet 0 starts a coherent read. The DCE0 in Chiplet 2 finds that AIU1 in Chiplet 0 is the owner and sends a snoop request. AIU1 has the data in UD state and sends a DtwReq to DMI0 in chiplet 2 and the DtrReq to AIU0 in Chiplet 0. There is no change in the way the Ncore units set the Initiator and Target ids except that they now carry more information.

In addition, note the update of the fields in GIU0 of chiplet 2 (highlighted in red) for which both the TargetId and the Did need to be updated to the local target. This cannot be done earlier because only the address map in

Chiplet2 has this information. Please refer to section 2.6 for the information on how the hierarchical address map works.

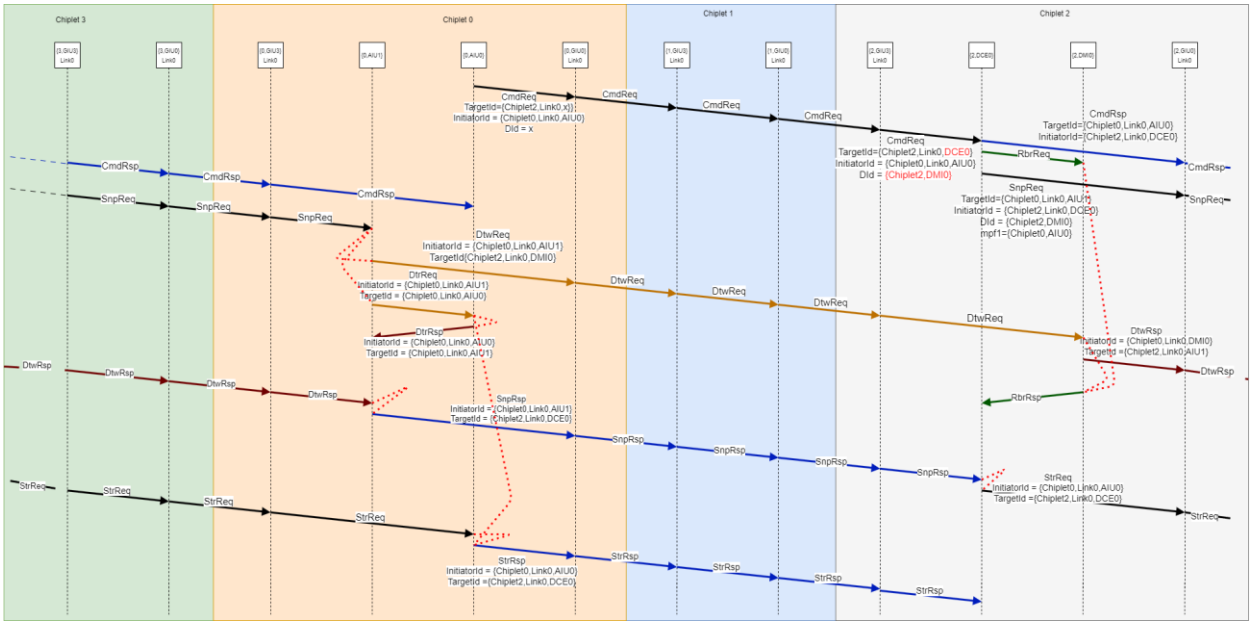


FIGURE 3-9 COHERENT READ FLOW WHERE AIUS ARE IN CHIPLET 0 AND DCE IS IN CHIPLET 2.

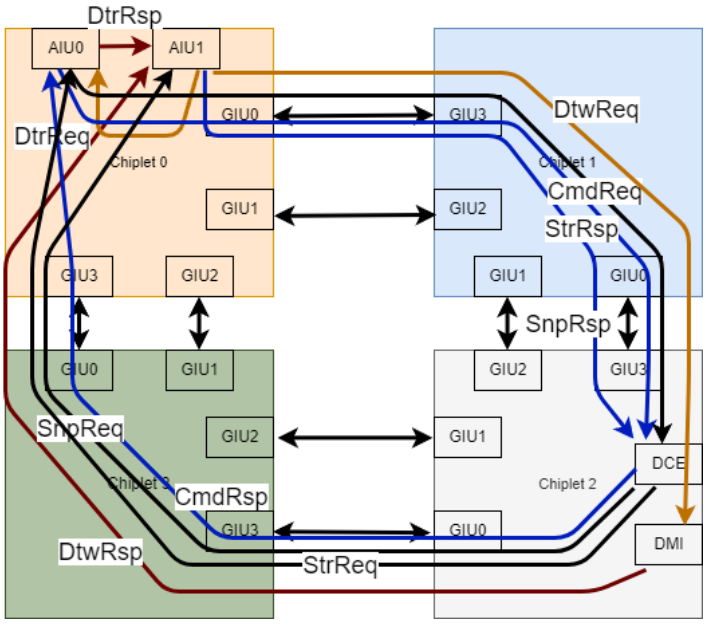


FIGURE 3-10 HIGHER LEVEL VIEW OF MESSAGE EXCHANGES OF FIGURE 3-9

4 Error handling and reporting

1149 The general behavior of error handling and reporting will follow a similar approach as the one currently used in Ncore, i.e., the unit where the error happen will log and raise an interrupt (if necessary) and will forward any preexisting error.

1152 In this chapter we will list the new errors which can happen in Ncore 3.8.

Error Name	Error Type	Enable	Interrupt	Mission Fault	Logged information	Action
Unknown RequesterGlobalId This happens when an AIU tries to attach but the Snoop filters are not configured yet.	Uncorrectable	SoftwareProgConfig	Yes, DCE	Yes, DCE	ErrType code: xC ErrInfo: [15:6] GlobalRequesterId	SysRsp CmStatus: {2'b01,ChipletId,1'b0,011}
Unknown Coherent Agent This happens when an AIU sends a CmdReq but the Snoop filters are not configured yet.	Uncorrectable	SoftwareProgConfig	Yes, DCE	Yes, DCE	ErrType code: 0xC ErrInfo: [3:0] 4'b0110: Unknown caching agent	StrReq CmStatus: 8'b10000100. Do not continue sending snoops, do not update the directory. The native interface will propagate in the response as a decode error.
CXS_TX/RX timeout	Uncorrectable	TimeoutErr	yes, GIU	Yes GIU	ErrType code: 0xC	The value for the timeout is 8 ms and if this timeout is reached it should stop the handshake.
CXS TX/RX Checktype error	Uncorrectable	IntfCheckErr	Yes GIU	Yes GIU	ErrType code: 0xD ErrInfo: TX: 5'b10000 RX: 5'b10001	Drop the transaction
Illegal start pointer	Uncorrectable	TransErrInt	Yes GIU	Yes GIU	ErrType code: 0x8 ErrInfo: 2'b10	Drop the flit.
(GIU) Transport SMI Prot Error	Uncorrectable	TransErrInt	Yes GIU	Yes GIU	ErrType code: 0x8 ErrInfo: 2'b01	Drop the flit. This is only raised on CmdReq and UpdReq which require an update of the target id and Did in the GIU. Other concerto messages are not decoded.

(GIU) Decode Error	Uncorrectable	SoftwareProgConfig	Yes GIU	No	ErrType code: 0x7 ErrInfo : same as other unit. New error info : 4'b0101 : Inconstistent ChipeltId	Drop the flit. This is to help debuging of address map setting across chiplets. Inconsistent ChipletId is raised when the address map decode in the GIU gives a ChipletId which is different compared to the one already present in the TargetId.

TABLE 4-1 NEW ERRORS RELATED TO D2D

Additionally, for the purpose of the transport error (ErrType code 0x8), every Ncore unit will log the ChipletId in ErrInfo[5:4] and the LinkId in ErrInfo[3:2]. This change will be reflected in the table of Ncore system architecture document.

5 FUSA

Each Chiplet/Die will have its own safety controller which will work independently from each other. No cross chiplet mechanism, synchronization or centralization is planned. It is the responsibility of the SOC integrator to implement a way to collect the alarms from multiple Ncore per its ASIL requirement.

The GIU will include a Safety controller which will need to be connected to the system safety controller, and it will behave similarly to any other Ncore unit.

In this version the GIU is considered as being part of the transport so the concerto messages going through it are protected by end-to-end protection. An architectural FMEDA needs to be done on the GIU to see if this safety mechanism is sufficient to guarantee a sufficient level of coverage. Depending on the result of this study, two approaches should be considered:

- Additional safety mechanisms to protect the state machine of the GIU.
- Full duplication of the entire units.

Although the preferred choice would be to avoid full duplication in both ASIL B and ASIL D systems, we will implement duplication for both ASIL B and ASIL D and depending on the result of the analysis will either remove the duplication or introduce safety mechanism to the state machine logic.

6 Trace and debug

No cross-die trace and debug is supported at this point. However, the GIU will instantiate a trace capture block and will log the messages seen at its SMI interface to the DVE. Its behavior will be identical to the rest of the Ncore units.

7 Initialization and boot-up

7.1 The system view of the initialization and boot up flow

The boot-up and initialization flow is dictated by the third party PHY provider. The flow can vary from one vendor to another, and it can vary from one PHY standard such as UCIe to another such as PCIe. There are some necessary IPs/modules from the SoC level to complete the initialization flow. Necessary software configurations and customizations are also required to achieve a more generic flow so to inter-operate with more than one third party controller IP provider.

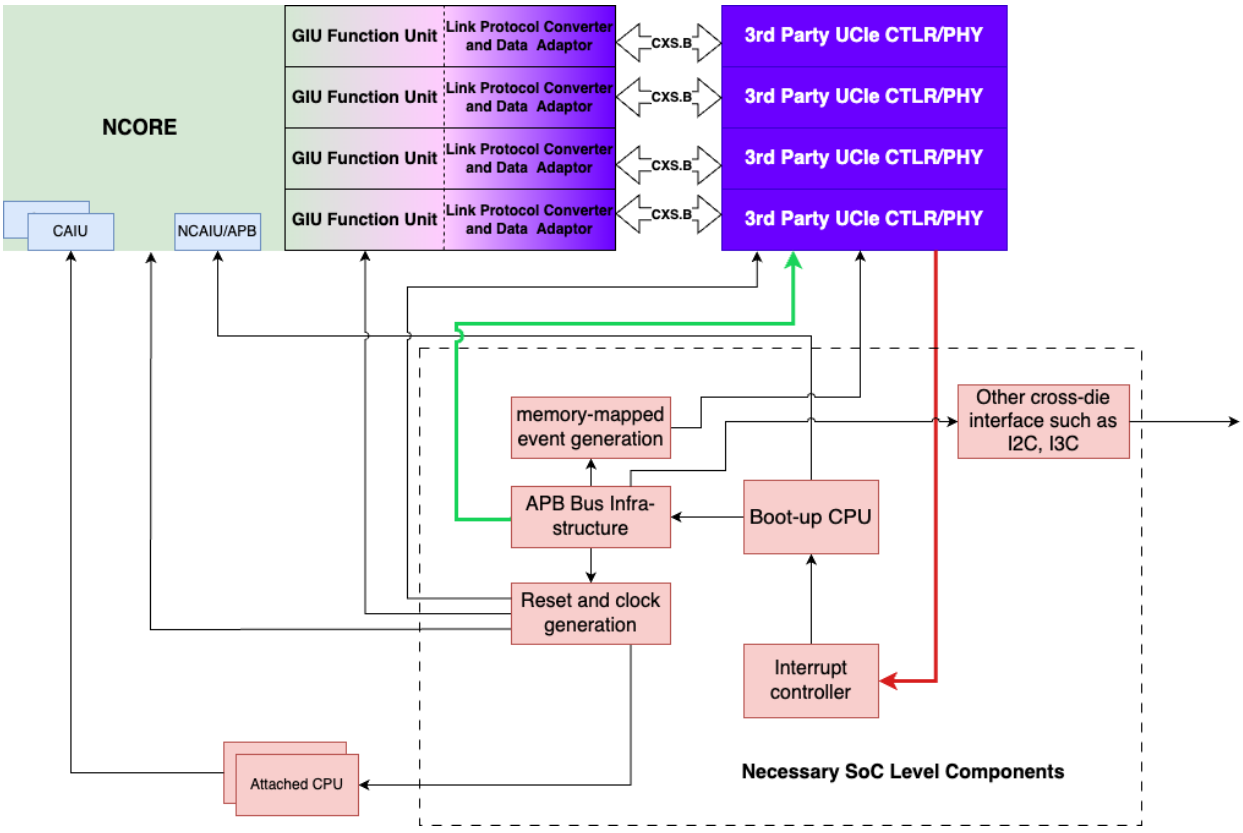


FIGURE 7-1 BOOT-UP SUBSYSTEM WITH SOC COLLATERALS INCLUDED

As shown in Figure 7-2, the following necessary SoC level modules/IPs should be scoped to work together with an Ncore to complete the initialization flow:

- CPU(s)
  - It is employed to manage the initialization flow, acknowledge relevant interrupts and drive necessary initialization events and signals, schedule the sequence of resets. Sometimes it is a standalone CPU which is used to manage the overall SoC boot-up sequence, it is called as the boot-up CPU in Figure 7-2. CPUs can also be attached to an Ncore as a coherent or non-coherent agent, sometimes one or more of this kind of CPU can also involve in some segments of boot-up sequence. However, they are still considered being managed by the boot-up CPU from the boot-up and initialization perspective.

- A bus fabric (or an APB infrastructure at very minimum)
  - Most of the 3rd party die-to-die controllers and PHY use an APB interface to carry out necessary customized initialization operations and sideband handshakes. Also, necessary controller and PHY CSR accesses are required by the boot-up CPU to schedule and sequence the initialization flow.

- An interrupt controller
  - Interrupts/events are required to trigger the boot-up CPU to acknowledge or initiate the initialization process.

- Other cross-die interface(s) such as I2C or I3C to facilitate boot-up sequence among multiple dies.
  - This type of interface is used to schedule or negotiate the handshake sequence between any two dies to make sure the boot-up sequence and handshake happens in the right order to get the physical link(s) set up properly.

- Necessary add-on registered mapped events:
    - Some necessary events are required to complete 3rd party link layer handshakes or exchange necessary parameters and configuration information. Relevant events should be driven by the memory mapped registers to generalize the overall boot-up flow.

As shown in Figure 7-2, the following interfaces among the multiple chiplets are expected to enable a smooth initialization for a given assembly:

- One or multiple pairs of UCIe mainband (shown as thick black double-arrow line) and UCIe sideband (shown as thinner red double-arrow line) must be present to carry necessary UCIe protocol related handshake between any two UCIe link partners.
- Extra interface such as I2C or I3C (shown as green thick double-arrow line) that provides necessary boot-up sequencing between and among multiple chiplets are very much required

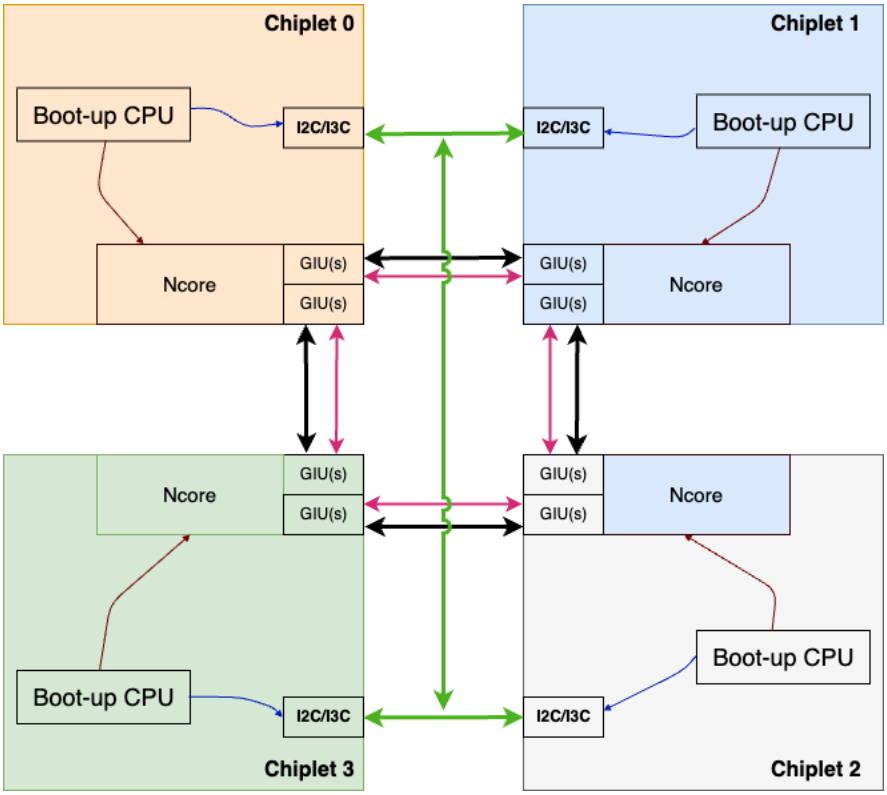


FIGURE 7-2 RELEVANT INTERFACES INVOLVED FOR THE BOOT-UP AND INITIALIZATION

7.2 Initialization and boot-up sequence

7.2.1 The general flow

New steps are added to the boot flow as described in the Reference manual. The below summarizes a general boot-up sequence. The steps below extend the boot sequence shown in the reference manual:

1. Perform CSR access and go through the Ncore component discovery described in Ncore Component Discovery.
2. Activate all the CXS links. To do so software needs to write a 1 to the field CXS_RX/TX_en of register in Table 7-1 GIUCXSLR : GIU CXS Link register. Address OFFSET: 0x50. The link is activated successfully when the CXS_RX_status/CXS_TX_status fields show that the CXS link is in the RUN state.

3. Configure DCEUCAMR [i] in every DCE (see section 3.7.3) to allocate every coherent agent to a Snoop filter.
4. Configure the register required for Domain attachment (see section 3.10)
5. Configure the registers required for DVM handling (see section 3.9)
6. Configure the registers required for Event handing (see section 3.11)
7. Configure required memory and peripheral address spaces as described in “address space” (see of the reference manual).
8. Configure required memory and peripheral address spaces as described in “Address space configuration” (see of the reference manual).
9. Configure error registers as needed to enable required error detection and reporting as described in *Error Detection and Reporting configuration* (see of the reference manual).
10. If preset, configure and enable all needed SMCs as described in “SMC configuration” (see of the reference manual).
11. If preset, configure and enable all needed Proxy caches as described in “Proxy cache configuration” (see of the reference manual).
12. Read register bit DCEUSFMAR[MntOpActv] to verify the DCE Snoop Filter RAM is initialized.
13. Enable/disable EVENTI and EVENTO port as described in *WFE Wakeup Event Signaling Support* if present with XAIUUTCR, CAIUUTCR, and DCEUTCR EventDisable field(see of the reference manual).
14. 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*(see of the reference manual).
15. Set up all the credit counter registers in the AIUs.
16. Optional: the timeout has already default value but depending on the traffic and design implementation, we recommend customizing correctly the timeout register value by step of 4,000 clocks cycles of:
    a. xAIUTOCR: transaction timeout value. In case of huge AXI burst or high traffic on many agents, we recommend reviewing this value.
    b. xAIUEHTOCR: XAIU Event Handshake Timeout value depending on your design reaction time.
    c. xAIUSEPTOCR & xAIUSCPTOCR are internal Ncore timeouts that do not need to be updated.
17. Optional: Set up quality of service QOS as described in *Quality of Service Support*.

It is important to note that for Ncore, the state of the CXS link is the state of the D2D link. Ncore has no knowledge of the underlying PHY/controller.

Some controllers, like the Synopsys one, may require that the RX and the TX be activated in a certain order and only from one side. It is the responsibility of the boot software to make sure that those conditions are fulfilled.

Last, in a configuration where not all chiplet are used this is the step at which some link would remain inactive.

7.2.2 Initialization and boot-up flow with Synopsys UCIe controller

To initialize a link with the Synopsys controller (step 2) the following steps need to happen in the correct order:

- Both sides need to write 1 to the CXS_RX_en field of CXSGIUCXLR (see section 7.2.3).
- One and only one side (we will call it initiator, and the other side will be called target) needs to assert the signal called app_lttsm_en going to the Synopsys controller (outside of Ncore). This can be guaranteed by the CPU exchanging information on an I2C bus or using a predefined software order based on the value of the ChipletId in GRBLSR.
- The target waits for ocxsb_irq indicating an active_request_irq to assert in order to assert app_lttsm_en.
- The initiator waits for ocxsb_irq to assert indicating a protocol_in_active_irq in order to write 1 to the CXS_TX_en field of CXSGIUCXLR (see section 7.2.3).
- The target waits for ocxsb_irq indicating protocol_in_active_irq to assert in order to write the CXS_TX_en field of CXSGIUCXLR (see section 7.2.3).
- Both sides need to poll the status of the CXS link until it is in active state

Once the link is in active state it is ready to use, and the boot can continue to the next steps.

Figure 7-3 Initialization Flow with Synopsys UCIe Controller Using CXS.B Interface is a snippet from Synopsys documentation which illustrate the handshake. Note that the custom flit only exists when the

controller is in CXL/CCIX mode.

Figure 4-12 UCIe to CXS Link Initialization Process

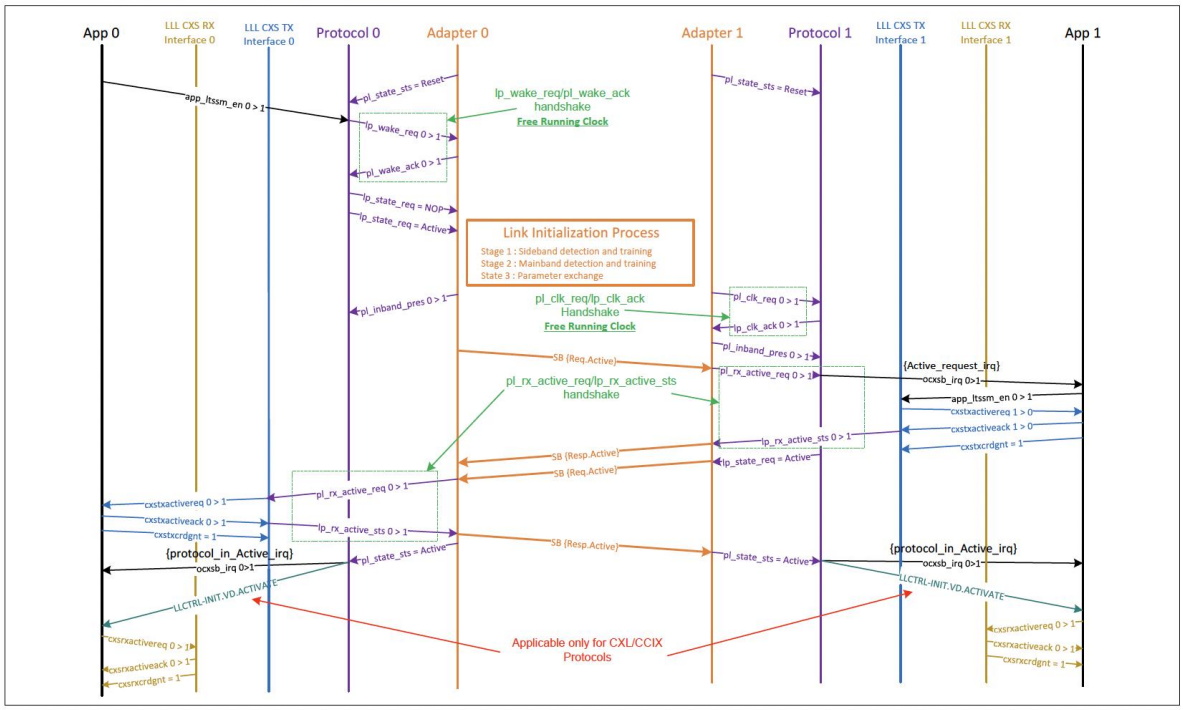
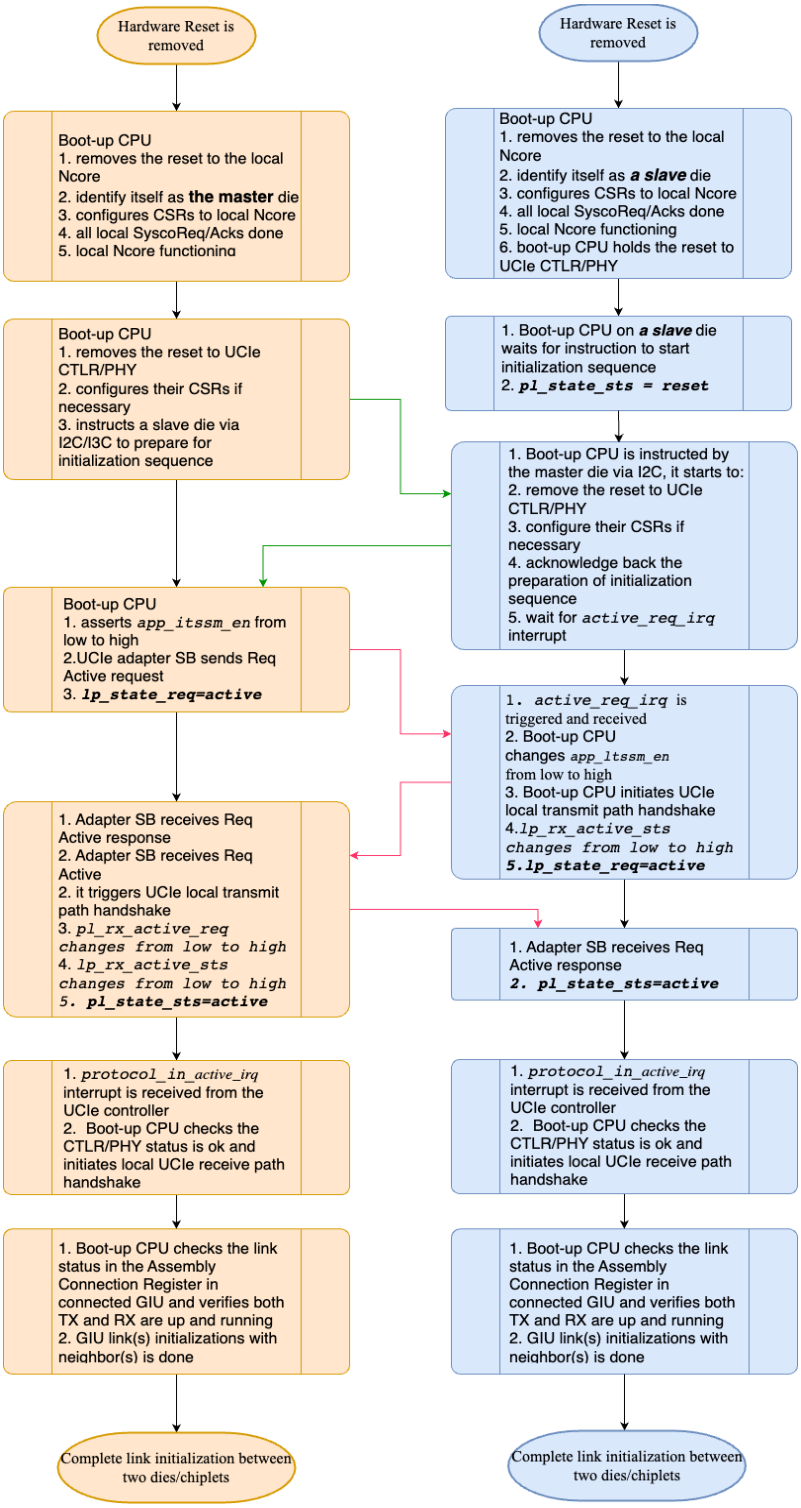


FIGURE 7-3 INITIALIZATION FLOW WITH SYNOPSYS UCIE CONTROLLER USING CXS.B INTERFACE

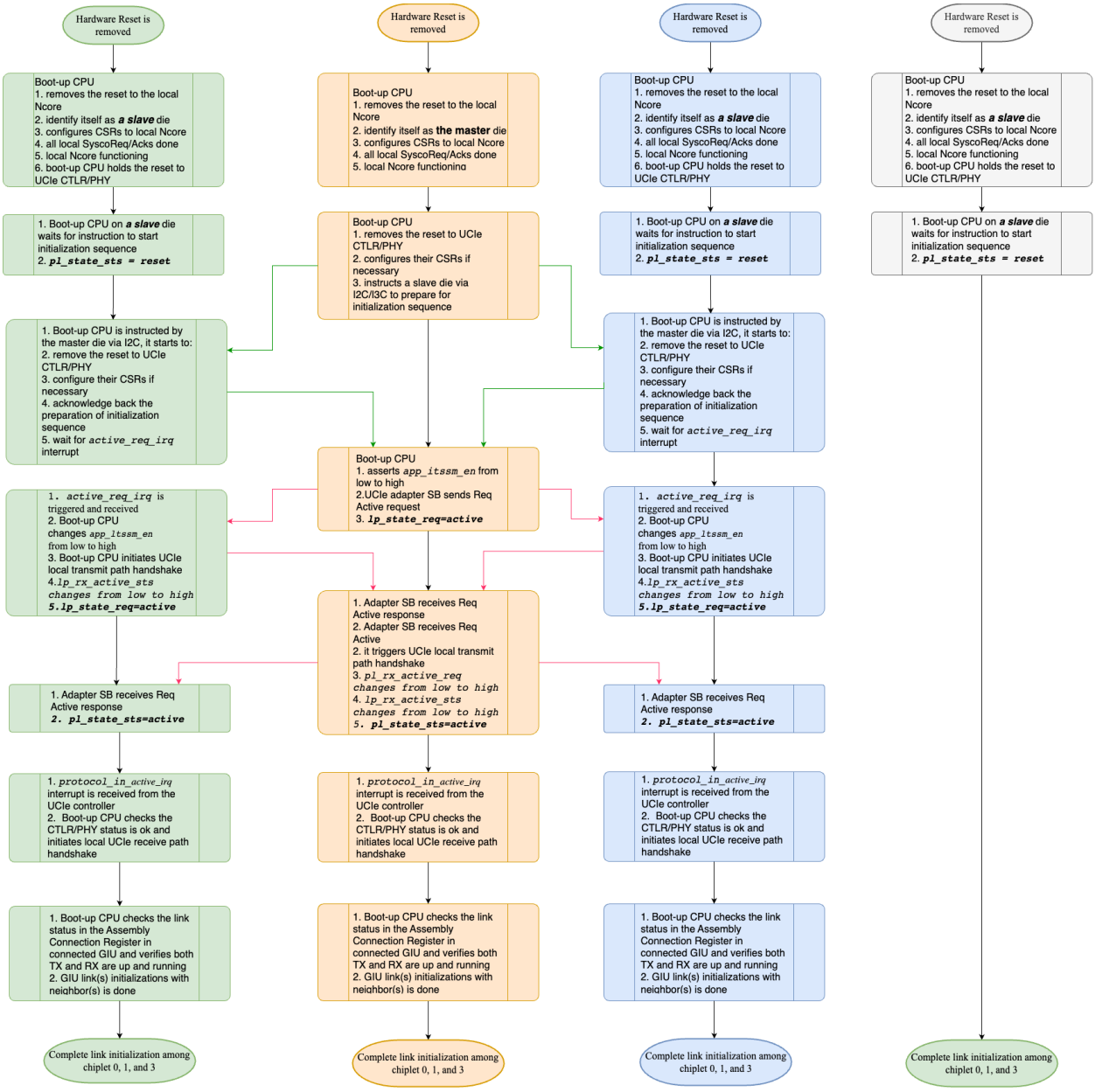
The boot-up and initialization sequence for an assembly with four chiplets is shown in Figure 7-5 and Figure 7-6:

1. In the first phase, for an assembly as shown in Figure 7-2, chiplet 0 initiates the initialization flow among chiplet 0, 1 and 3 as shown in Figure 7-5
2. Once all the links that directly connected among chiplet 0, 1 and 3 have finished the initialization flow, chiplet 0 needs to instruct the boot-up CPU on chiplet 2 via the I2C interface or via the user defined messaging field defined in Table 7-8 GRBLSR GRB Link Status Register. Address offset: xxx.
3. The boot-up CPU on chiplet 2 then needs to initiate a similar initialization flow as shown in Figure 7-6.
4. Once the above steps are completed, any AIU can start coherent or DVM attachment process.
5. The normal operations start hereafter.



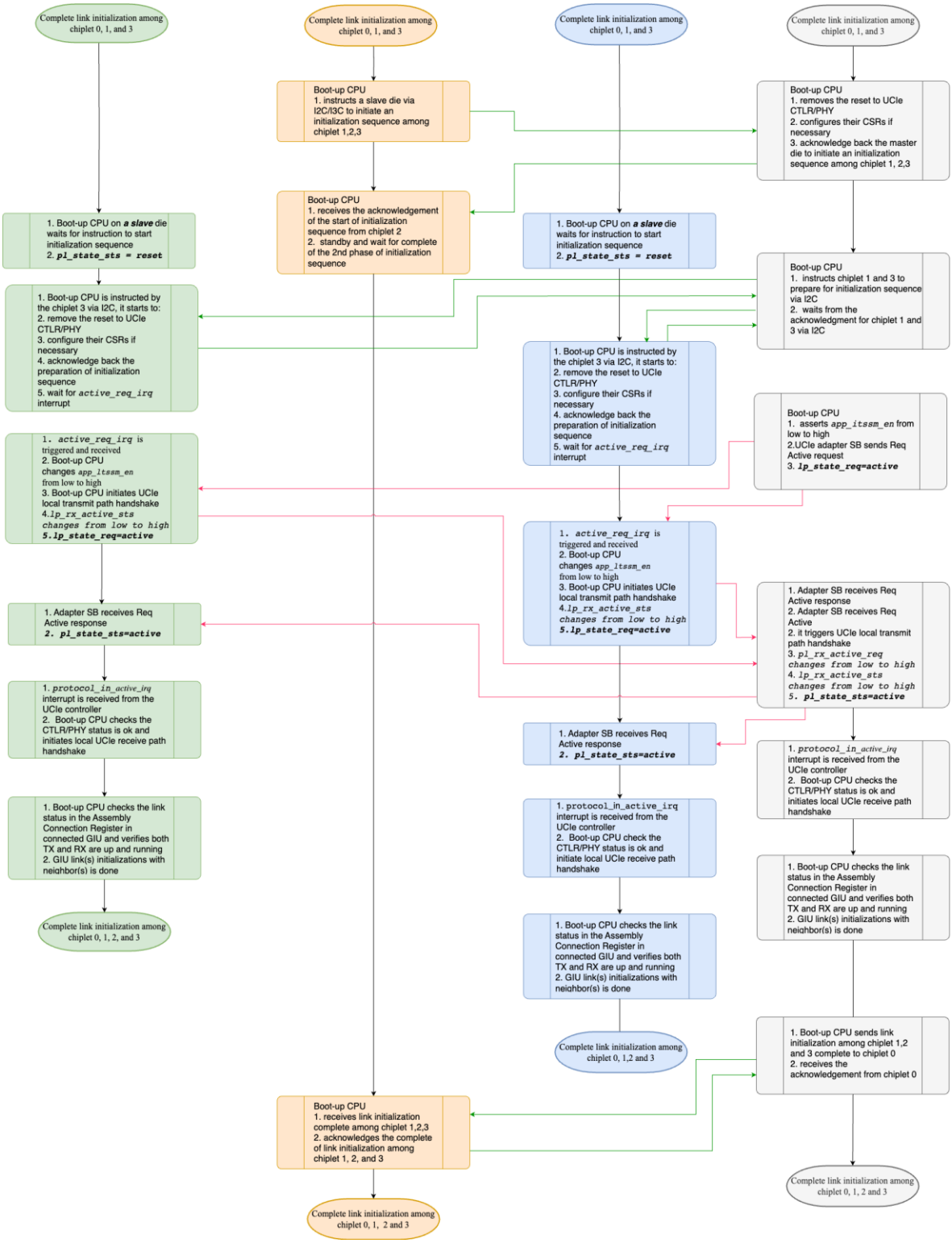
1314

FIGURE 7-4 THE FLOWCHART SHOWS THE HANDSHAKE PROCESS WITH AN ASSEMBLY OF TWO CHIPLETS



1317

FIGURE 7-5 THE FIRST PHASE OF THE INITIALIZATION FLOW WITH AN ASSEMBLY OF FOUR CHIPLETS



1320

FIGURE 7-6 THE SECOND PHASE OF THE INITIALIZATION FLOW WITH AN ASSEMBLY OF FOUR CHIPLETS

7.2.3 GIUCXSLR: GIU CXS Link register.

This register is used to control the state of the CXS link and whether the GIU will initiate/respond to a handshake on its CXS TX and RX interfaces.

Bits	Field Name	SW access	HW access	Reset value	Descriptions
0	CXS_RX_en	RW	RO	0x0	If this bit is set to one while he CXS RX link is in STOP state, it will allow the GIU to respond to an activation request from the CXS initiator this GIU is connected to. If this bit is set to 0 while in RUN state will trigger a return of all the link credit to the transmitter and the assertion of a deactivation hint to the transmitter.
2:1	CXS_RX_status	RO	RW	0x0	This register represents the status of the CXS TX link. 0x0: STOP state. 0x1: ACTIVATE state. 0x2: RUN state. 0x3: DEACTIVATE state
3	CXS_RX_deactivation_Error	RO	RW	0x0	This bit asserts if the CXS initiator that this GIU is connected to did not respond before a set timeout is reached.
4	CXS_TX_en	RW	RW	0x0	If this bit is set to one while the CXS_TX link is in STOP state it will start an activation request to the CXS receiver this GIU is connected to. Hardware may also set this register back to 0 when accepting a deactivation hint.
6:5	CXS_TX_status	RO	RW	0x0	This register represents the status of the CXS RX link. 0x0: STOP state. 0x1: ACTIVATE state. 0x2: RUN state. 0x3: DEACTIVATE state
7	CXS_TX_activation_Error	RO	RW	0x0	This bit asserts if the CXS receiver that this GIU is connected to did not respond before a set timeout is reached.
8	CXS_TX_REFUSE_DEACTHINT	RW	RO	0x0	Set this bit to one to allow the GIU to refuse the deactivation Hint from the CXS master to which it is connected. If this register is not set it will allow the GIU to respond to a deactivation hint. Doing so will update the CXS_TX_en register to 1.
31:10	Reserved	RO	RO	0x0	Reserved

TABLE 7-1 GIUCXSLR : GIU CXS LINK REGISTER. ADDRESS OFFSET: 0X50.

7.2.4 GIUUIDRGIU: GIU Identification register

Bits	Field Name	SW access	HW access	Reset value	Descriptions
7:0	RPN	RO	RO	rpn	GIU register page Number (within its NRR)
11:8	NRRI	RO	RO	nrri	Identifier of the Ncore 3 Register Region\nin which this GIU resides
16:12	NUnitId	RO	RO	nUnitId	Ncore3 GIU Unit identifier
18:17	ConnectedChipletId	RW	RO	0x0	ConnectedChipletID is the Chiplet that this GIU connects to. This needs to be programmed at boot according to the configuration the chip is booting in.
19	Reserved	RO	RO	0x0	Reserved
24:20	ConnectedNUnitId	RW	RO	0x0	ConnectedNUnitID is the NUnitId of the remote GIU that this GIU is connected to. This needs to be programmed at boot according to the configuration the chip is booting in.
28:25	Reserved	RO	RO	0x0	Reserved
30:29	LinkId	RW	RO	0x0	LinkId assigned to this GIU. This needs to be programmed at boot according to the configuration the chip is booting in.
31	Valid	RO	RO	0x1	Value of 1 validates this register. This bit is set to 1 if the GIU is implemented.

TABLE 7-2 GIUUIDRGIU IDENTIFICATION REGISTER, ADDRESS OFFSET : 0X0

This register exists in every Ncore unit and is extended in the GIU to contain information about which GIU it is connected to. Table 7-2 GIUUIDRGIU Identification register, Address offset : 0x0 shows the definition of an assembly register per physical GIU link. This register should be located inside of each GIU. Please note that only the chiplet that has the direct connection to the current GIU/chiplet is specified during the build time and is initialized by the boot-up CPU as the first step during the boot-up process.

7.2.5 GRBTIR: GRB Topology Information register

Table 7-3 GRBTIR: GRB Topology Information register is the register that represents the link connections for a specific assembly. Each of the four bits represents one GIU link if the current chiplet has connection(s) to it. Please note that the lower half of the register represents direct connections between the current chiplet and its direct neighbor(s).

The upper half of the register represents indirect connections, which the current chiplet must go through an intermediate chiplet to reach the other chiplet at the end.

This register is populated after reset by Hardware from a set of tie offs based on which chiplet this GRB is in. The structure of the Tie Off is TBD.

1350

Bits	Field Name	SW access	HW access	Reset value	Descriptions
3:0	DirectlyToChiplet0	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 0 directly. If the current chiplet is the chiplet 0 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
7:4	DirectlyToChiplet1	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 0 directly. If the current chiplet is the chiplet 0 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
11:8	DirectlyToChiplet2	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 0 directly. If the current chiplet is the chiplet 0 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
15:12	DirectlyToChiplet3	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 0 directly. If the current chiplet is the chiplet 0 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
19:16	IndirectlyToChiplet0	RW	RO	0x0	3:0: each bit represents a GIU:

Bits	Field Name	SW access	HW access	Reset value	Descriptions
					1'b1: implies the corresponding GIU has a path to connect to chiplet 3 indirectly. If the current chiplet is the chiplet 3 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
23:20	IndirectlyToChiplet1	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 3 indirectly. If the current chiplet is the chiplet 3 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
27:24	IndirectlyToChiplet2	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 3 indirectly. If the current chiplet is the chiplet 3 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.
31:28	IndirectlyToChiplet3	RW	RO	0x0	3:0: each bit represents a GIU: 1'b1: implies the corresponding GIU has a path to connect to chiplet 3 indirectly. If the current chiplet is the chiplet 3 for an assembly, 1'b1 indicates the corresponding GIU is used in the current assembly. The index used for that vector is the NUnitId of the GIU. It is loaded from maestro tie offs based on the ChipletId this GRB is in.

TABLE 7-3 GRBTIR: GRB TOPOLOGY INFORMATION REGISTER

7.2.6 Example of GRBTIR programming.

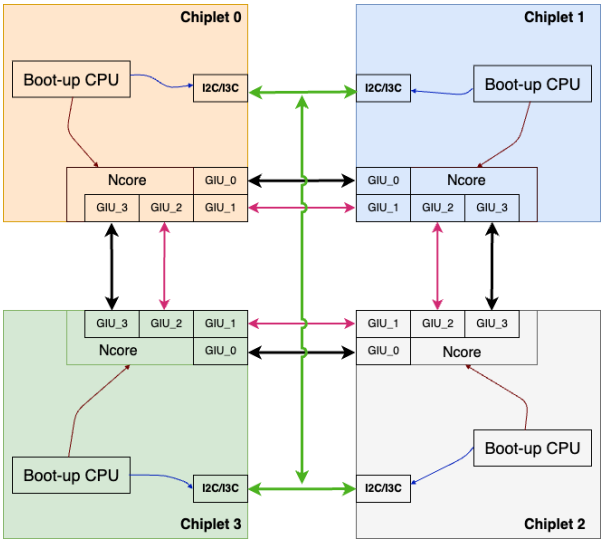


FIGURE 7-7 LINK CONFIGURATIONS ACROSS CHIPLETS

Bits	Field Name	Descriptions
3:0	DirectlyToChiplet0	4'b1111 Current chiplet ID is 0, and ALL four links are used
7:4	DirectlyToChiplet1	4'b0011 It has direct links as GIU 0 and 1 with chiplet 1
11:8	DirectlyToChiplet2	4'b0000 No direct link(s) to chiplet 2
15:12	DirectlyToChiplet3	4'b1100 It has direct links as GIU 2 and 3 with chiplet 3
19:16	IndirectlyToChiplet0	4'b0000 Don't care since the current chiplet is chiplet 0
23:20	IndirectlyToChiplet1	4'b0000 It has direct connections with chiplet 1; thus, this field should be zeros
27:24	IndirectlyToChiplet2	4'b0011 Indicates it connects to chiplet 2 indirectly with its GIU 0
31:28	IndirectlyToChiplet3	4'b0000 It has direct connections with chiplet 3; thus, this field should be zeros

TABLE 7-4 THE LINK CONFIGURATIONS FOR CHIPLET 0 IN AN ASSEMBLY SHOWN IN FIGURE 7-7

Bits	Field Name	Descriptions
3:0	DirectlyToChiplet0	4'b0011 It has direct links as GIU 0 and 1 with chiplet 0
7:4	DirectlyToChiplet1	4'b1111 Current chiplet ID is 1, and ALL four links are used
11:8	DirectlyToChiplet2	4'b1100 It has direct links as GIU 2 and 3 with chiplet 2
15:12	DirectlyToChiplet3	4'b0000 No direct link(s) to chiplet 3
19:16	IndirectlyToChiplet0	4'b0000 It has direct connections with chiplet 0; thus, this field should be zeros
23:20	IndirectlyToChiplet1	4'b0000 Don't care since the current chiplet is chiplet 1
27:24	IndirectlyToChiplet2	4'b0000 It has direct connections with chiplet 2; thus, this field should be zeros
31:28	IndirectlyToChiplet3	4'b0011 Indicates it connects to chiplet 3 indirectly with its GIU 0 and 1

TABLE 7-5 THE LINK CONFIGURATIONS FOR CHIPLET 1 IN AN ASSEMBLY SHOWN IN FIGURE 7-7

Bits	Field Name	Descriptions
3:0	DirectlyToChiplet0	4'b0000 No direct link(s) to chiplet 0
7:4	DirectlyToChiplet1	4'b1100 It has direct links as GIU 2 and 3 with chiplet 1
11:8	DirectlyToChiplet2	4'b1111 Current chiplet ID is 2, and ALL four links are used
15:12	DirectlyToChiplet3	4'b0011 It has direct links as GIU 0 and 1 with chiplet 3
19:16	IndirectlyToChiplet0	4'b0011 Indicates it connects to chiplet 0 indirectly with its GIU 0 and 1
23:20	IndirectlyToChiplet1	4'b0000 It has direct connections with chiplet 1; thus, this field should be zeros
27:24	IndirectlyToChiplet2	4'b0000 Don't care since the current chiplet is chiplet 2
31:28	IndirectlyToChiplet3	4'b0000 It has direct connections with chiplet 3; thus, this field should be zeros

TABLE 7-6 THE LINK CONFIGURATIONS FOR CHIPLET 2 IN AN ASSEMBLY SHOWN IN FIGURE 7-7

Bits	Field Name	Descriptions
3:0	DirectlyToChiplet0	4'b1100 It has direct links as GIU 2 and 3 with chiplet 0
7:4	DirectlyToChiplet1	4'b0000 No direct link(s) to chiplet 1
11:8	DirectlyToChiplet2	4'b0011 It has direct links as GIU 0 and 1 with chiplet 2
15:12	DirectlyToChiplet3	4'b1111 Current chiplet ID is 3, and ALL four links are used
19:16	IndirectlyToChiplet0	4'b0000 It has direct connections with chiplet 0; thus, this field should be zeros
23:20	IndirectlyToChiplet1	4'b0011 Indicates it connects to chiplet 1 indirectly with its GIU 0 and 1
27:24	IndirectlyToChiplet2	4'b0000 It has direct connections with chiplet 2; thus, this field should be zeros
31:28	IndirectlyToChiplet3	4'b0000 Don't care since the current chiplet is chiplet 3

TABLE 7-7 THE LINK CONFIGURATIONS FOR CHIPLET 3 IN AN ASSEMBLY SHOWN IN FIGURE 7-7

7.2.7 GRBLSR: GRB Link Status register

Table 7-8 is the register that controls the GIU link(s) and tracks its status. It is updated by a CPU based on the value read in the GIU register of section 7.2.3, Table 7-1 GIUCXSLR : GIU CXS Link register. Address OFFSET: 0x50.

The boot-up CPU updates and compares the links chiplet by chiplet across all configured connections for a specific assembly as defined in Table 7-3 GRBTIR: GRB Topology Information register. The boot-up CPU sets the corresponding DirectlyToChiplet<x>Status status to 1'b1 between any two chiplets once the corresponding 4-bit link connections are matched between Table 7-3 GRBTIR: GRB Topology Information register and Table 7-7 The Link Configurations for Chiplet 3 in An Assembly Shown in Figure 7-7.

Bits	Field Name	SW access	HW access	Reset value	Descriptions
3:0	DirectlyToChiplet0GIUStatus	RW	RO	0x0	3:0: each bit represents a GIU. GIU are indexed by their NUnitId. 1'b1: implies the corresponding GIU is up and running
4	DirectlyToChiplet0Status	RW	RO	0x0	1'b1 implies that ALL link(s) to die/chiplet 3 is up and running. 1'b0 is NOT. Reset value is 1'b0
8:5	DirectlyToChiplet1GIUStatus	RW	RO	0x0	3:0: each bit represents a GIU. GIU are indexed by their NUnitId. 1'b1: implies the corresponding GIU is up and running
9	DirectlyToChiplet1Status	RW	RO	0x0	1'b1 implies that ALL link(s) to die/chiplet 3 is up and running. 1'b0 is NOT. Reset value is 1'b0
13:10	DirectlyToChiplet2GIUStatus	RW	RO	0x0	3:0: each bit represents a GIU. GIU are indexed by their NUnitId. 1'b1: implies the corresponding GIU is up and running
14	DirectlyToChiplet2Status	RW	RO	0x0	1'b1 implies that ALL link(s) to die/chiplet 3 is up and running. 1'b0 is NOT. Reset value is 1'b0
18:15	DirectlyToChiplet3GIUStatus	RW	RO	0x0	3:0: each bit represents a GIU. GIU are indexed by their NUnitId. 1'b1: implies the corresponding GIU is up and running
19	DirectlyToChiplet3Status	RW	RO	0x0	1'b1 implies that ALL link(s) to die/chiplet 3 is up and running. 1'b0 is NOT. Reset value is 1'b0
21:20	ChipletId	RO	RW	0x0	ChipletId this register resides in. It is loaded after reset by hardware from a tie-off.
31:22	UserDefinedMessaging	RW	RO	0x0	This register is meant to be used by configuration software to exchange information between CPU on different chiplet without having to access memory. In particular it can be used to pass the ownership of link Initialization to a different chiplet when the topology is not fully connected.

TABLE 7-8 GRBLSR GRB LINK STATUS REGISTER. ADDRESS OFFSET: XXX

Please note that the UserDefinedMessaging subfield in Table 7-8 GRBLSR GRB Link Status Register. Address offset: xxx. is reserved for the information exchange during the boot-up or other means to facilitate cross chiplet communications if needed. Any CPU who has the access to the CSR space from any Ncore instance across an assembly can read and write to this subfield in this register, extra synchronization and reserved usage to the subfield is expected and managed beyond Ncore level.

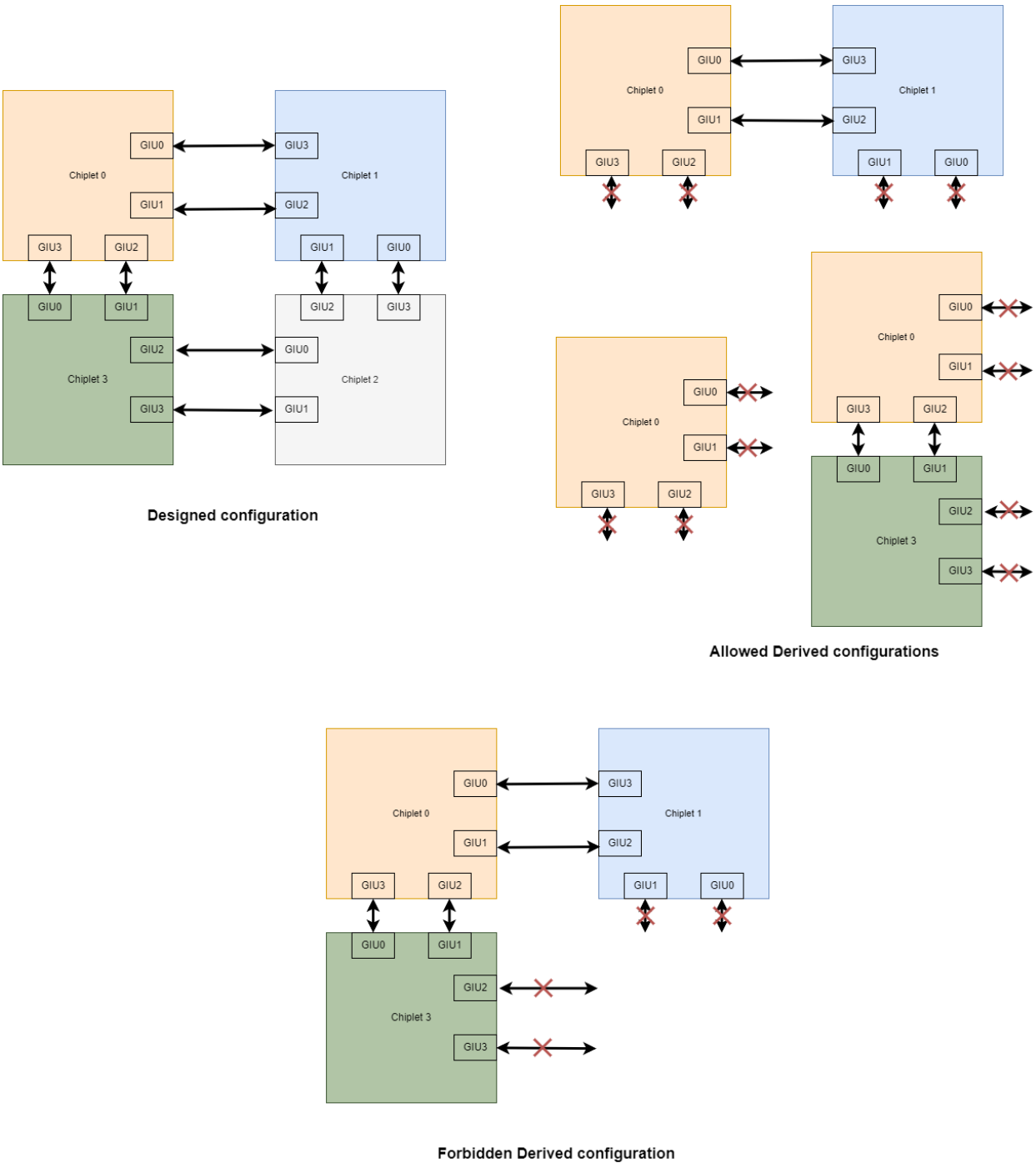
8 Example of partially populated chip use.

8.1 Introduction

For Ncore 3.8.0 only one assembly can be designed in Maestro. However, it will be allowed to use a subset of that configuration. Here is an extensive list of derived configurations which will be allowed:

- For a 2x2 mesh: using one die by itself, using two dies which are directly connected. 3 dies are not allowed.
- For a 4x1/3x1 mesh: using any number of dies so long as connectivity is preserved becoming 3x1,2x1 mesh.
- For a 2,3,4 fully connected configuration: can use any number of dies.

It is important to note that a D2D link which used to connect to a Die that was removed cannot be remapped and will become unused. This is mostly a mechanism meant to do binning in case a chiplet fails in the package.



1398

FIGURE 8-1 EXAMPLES OF ALLOWED DERIVED CONFIGURATION AND FORBIDDEN DERIVED CONFIGURATION FOR A 2X2 MESH.

1401 Figure 8-1 illustrates how a 2x2 mesh of dies can be used in a single die or a two dies configuration. The arrows which are crossed red indicate that the die that it is to connect to is either absent, powered down or non-functioning.

The way to go from the derived configuration to another allowed derived configuration is to update the value of certain register which we will list in the next paragraph.

8.2 Recipe to program the system into a derived configuration

For the sake of simplicity, we will consider the system defined in Figure 8-1 and we will provide the value of the register for both the starting system and the derived system composed of chiplet 0 and 1.

Register name	Field/Value in the designed configuration	Field/Value in the derived configuration
DVEUCAER (Table 3-9)	Chiplet0CohEn =1'b1 Chiplet1CohEn =1'b1 Chiplet2CohEn =1'b1 Chiplet3CohEn =1'b1 CurrentConfigNumber = 3'b000	Chiplet0CohEn =1'b1 Chiplet1CohEn =1'b1 Chiplet2CohEn =1'b0 Chiplet3CohEn =1'b0 CurrentConfigNumber = 3'b001
DVEURSEE (Table 3-11)	Chiplet0EventEn = 1'b1 Chiplet1EventEn = 1'b1 Chiplet2EventEn = 1'b1 Chiplet3EventEn = 1'b1 CurrentConfigNumber = 3'b000	Chiplet0EventEn = 1'b1 Chiplet1EventEn = 1'b1 Chiplet2EventEn = 1'b0 Chiplet3EventEn = 1'b0 CurrentConfigNumber = 3'b001
DVEUDRSER (Table 3-6)	Chiplet0DVMEn = 1'b1 Chiplet1DVMEn = 1'b1 Chiplet2DVMEn = 1'b1 Chiplet3DVMEn = 1'b1 CurrentConfigNumber = 3'b000	Chiplet0DVMEn = 1'b1 Chiplet1DVMEn = 1'b1 Chiplet2DVMEn = 1'b0 Chiplet3DVMEn = 1'b0 CurrentConfigNumber = 3'b001
GIUCXSLR (Table 7-1)	All dies/all GIUs: CXS_RX_en = 1'b1 CXS_TX_en = 1'b1	GIU0 and 1 of Chiplet 0: CXS_RX_en = 1'b1 CXS_TX_en = 1'b1 GIU2 and 3 of Chiplet 0: CXS_RX_en = 1'b0 CXS_TX_en = 1'b0 GIU2 and 3 of Chiplet 1: CXS_RX_en = 1'b1 CXS_TX_en = 1'b1 GIU0 and 1 of Chiplet 1: CXS_RX_en = 1'b0 CXS_TX_en = 1'b0

TABLE 8-1 PROGRAMING OF THE RELEVANT REGISTERS IN A DERIVED CONFIGURATION OF 2 DIES

Additionally, the address map (GPRAR) must not refer chiplet 2 and chiplet 3 in the HUI field in the derived configuration. In case the address map refers to dies that do not exist those transactions will be sent to the corresponding GIU. Consequently, there is a risk that those transaction overflow in the network and block it.

Eventually, the transactions will timeout, but it might be too late for the network to not be completely deadlocked. In such eventuality the only access to the csr space will be through the debug port.

It is recommended but not mandatory to invalidate the caching agents that belong to a chiplet that do not exist in DCEUCAMR[i] (Table 3-2).

9 Low power management and control

9.1 Low-power mode for UCIe controller

Currently, CXS.B bridge of the Synopsys UCIe controller doesn't support low-power modes.[1]

10 Cross Die/Chiplet interrupt handling

Ncore 3.8 provides support to ARM GIC-700 users for cross die/chiplet communications. As shown in Figure 10-2, ARM GIC-700 has cross die interface that can be configured using ACE5-Lite protocol. To enable the cross-die interface, the ACE5-Lite manager interface needs to be connected with one of NCAIUs of an Ncore from the local die/chiplet; The subordinate ACE5-Lite interface of the GIC-700 distributor on the remote die/chiplet needs to be connected to a DII.

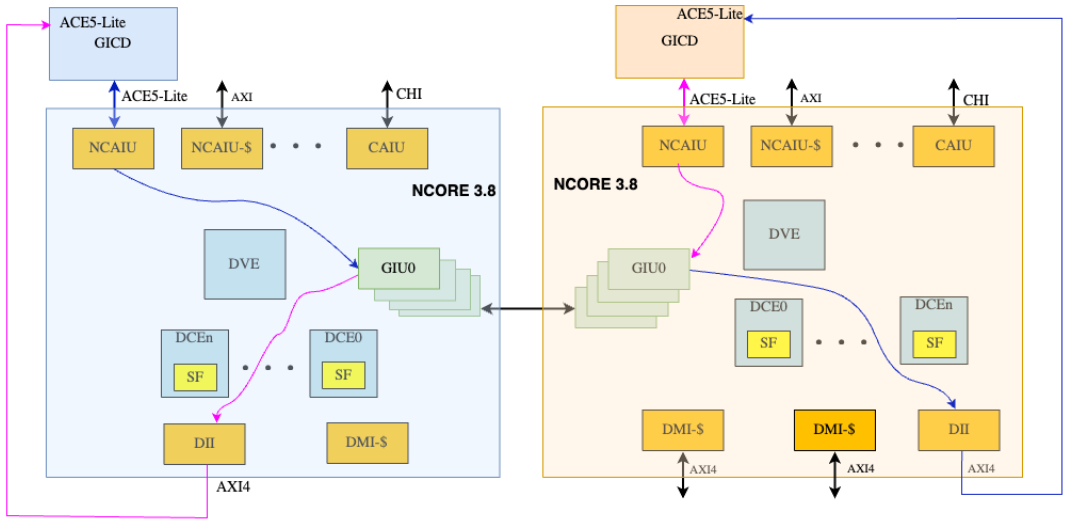


FIGURE 10-1 CROSS DIE/CHIPLET SUPPORT FOR ARM GIC-700

[1] As stated in section 4.7.2 in [12]

Since an Ncore DII can only support AXI4 interface, the awatop_s, a<x>domain>_s, a<x>snoop_s signals must be tied LOW when it connects to an ACE5_Lite subordinate interface of a GIC-700 distributor.[2]

Figure 3-1: GIC-700 Distributor

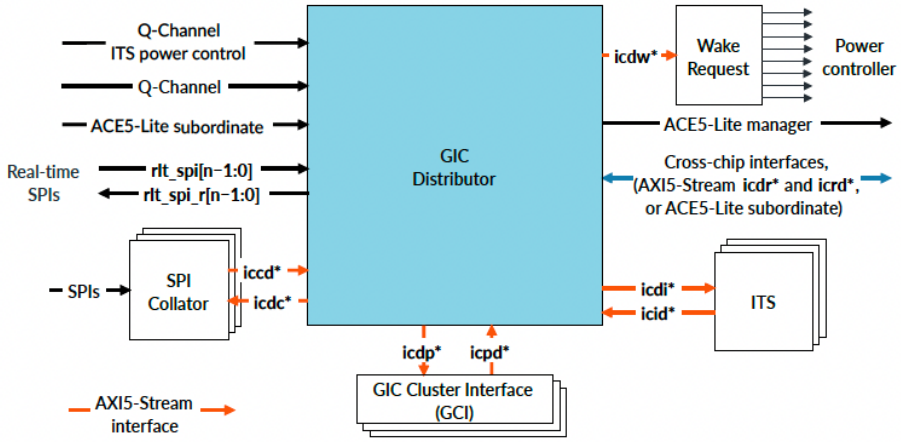


FIGURE 10-2 INTERFACES EMPLOYED IN ARM GIC-700 DISTRIBUTOR

[2] As described in section 3.1.2.2 and B.11 of [6]

11 Testing and debugging considerations

With the inclusion of the 3rd party controller and PHY in the data path, it is essential to have some testing and debugging capabilities to identify which segment of the data-path malfunctions or is mis-configured. One basic function is the loopback testing. Loopbacks normally can be carried directly from 3rd party PHY during the BIST test. However, the loopbacks discussed below focuses on the involvement of GIU to fulfil loopbacks with the mission mode settings.

11.1 Loopback functions

As shown in Figure 11-1, multiple loopback functions are supported by 3rd party PHYs for both standard and advanced packages[3].

11.1.1 Near-end loopback

Near-end loopback means to loopback the transmit data back as the receive data before it leaves the local chiplet. There are two kinds of near-end loopback available:

- Near-end loopback within a GIU. This is loopback within a specific GIU as shown as the purple dashed line in Figure 11-1, in such as a loopback mode, the loopback path is set up right before the CXS.B TX interface and is connected right after the CXS.B RX interface.
    - This is the loopback mode doesn't involve any 3rd party controller and PHY, it is served to make sure if a GIU is functioning or not.
    - This loopback mode shall cover as much as possible of common paths between the loopback mode and mission mode, the further it is located toward to an Ncore, the better.
- The 2nd category near-end loopback involves 3rd party controller and PHY. As shown in Figure 11-1, two options are available for this kind of near-end loopback, one option is to loopback data before the transceiver as shown in the red color dashed line and the other option is to loopback data after the transceiver as shown in the green color dashed line on Chiplet A.
    - When near-end loopback mode is enabled, the datapath between the two chiplets is broken at the line for which a specific near-end loop back is configured and there isn't any data traffic expected to be received in the corresponding remote Chiplet.
        - In this case, if the near-end loopback is configured in Chiplet A, no functional traffic is expected to receive at the remote die, which is Chiplet B in this case.

[3] Details can be found in [7] [8]

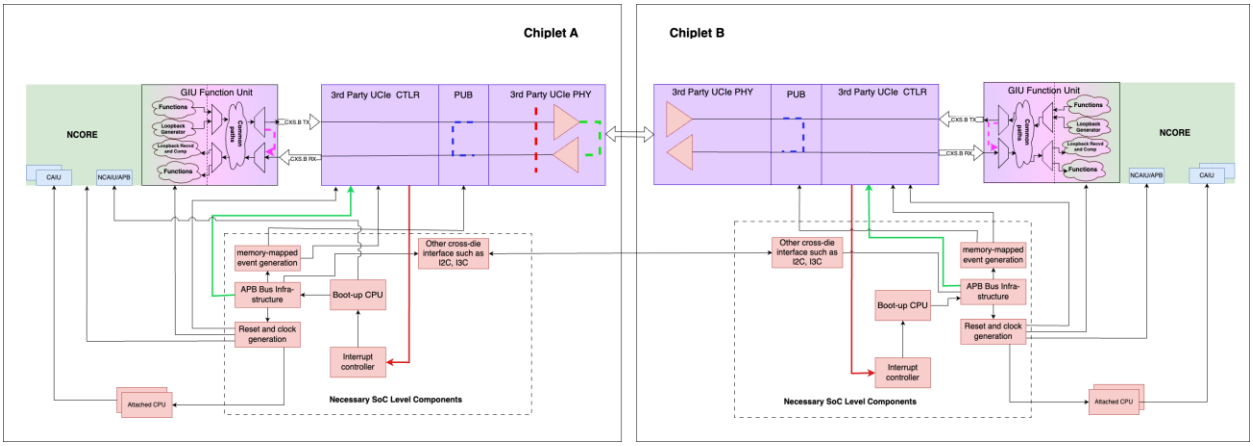


FIGURE 11-1 LOOPBACK SCENARIOS

11.1.2 Far-end loopback

For far-end loopback from chiplet A perspective, the data is loopback at the PUB of the chiplet B as shown in the blue color dashed line.

When far-end loopback is enabled, the PUB in the far-end needs to be configured with far-end loopback while the GIU in the near-end is configured as the loopback mode so the data path can be loopback from the PUB located at the far-end as shown as the blue dashed line in Figure 11-1. The 3rd party controller and PHY located in local die (chiplet A) are expected to work in a mission mode.

11.1.3 Controls and handlings of loopbacks

A few registers need to be added in every GIU module to fulfil the control and handling of the loopback functionalities:

- One is the loopback control and status register.
- The data seed register: one for the seed
- Sixteen mismatch data registers which record the value of the 64B flit which mismatched.

Bits	Field Name	SW Access	HW Access	Descriptions
0	LoopBackEn	RW	RO	1'b0: functional mode

Bits	Field Name	SW Access	HW Access	Descriptions
				1'b1: loopback mode Reset value 1'b0, default at the functional mode. In loopback mode the GIU will generate test flits on the CXS interface and will stop processing requests from Ncore.
1	NearLoopEn	RW	RO	1'b0: loopback externally to a GIU 1'b1: loopback internally within a GIU. The outgoing CXS flits looped back inside the GIU and are note propagated to the controller
2	FarLoopEn	RW	RO	1'b0: loopback externally to a GIU 1'b1: loopback internally within a GIU. The incoming CXS flits are looped back to the controller and are never propagated inside the GIU.
[5:4]	ShiftStart	RW	RO	2'b00: shifting to left by one bit 2'b01: shifting to left by 4 bits with wrap-around 2'b01: shifting to left by 8 bits 2'b10: shifting to left by 16 bits Reset value 2'b00, default to use shifting by one
[7:6]	DelayCounter	RW	RO	This field has two functions: if the loopback is set as single issue mode, this field is served as a delay counter for a group of loopback flits to be sent after the loopback is enabled; if the loopback is set as repetitive mode, the same delay counter is also served as the idle/delay counter before the next group of loopback flits is scheduled to send from a GIU. 2'b00: delayed by 16 GIU clock cycles 2'b01: delayed by 32 GIU clock cycles 2'b10: delayed by 64 GIU clock cycles 2'b11: delayed by 128 GIU clock cycles Reset value 2'b00
[9:8]	NumberOfTest	RW	RO	This field only has effects when the loopback is set as in repetitive mode, it is ignored otherwise. 2'b00: The loopback test repeats forever till it is stopped

Bits	Field Name	SW Access	HW Access	Descriptions
				2'b01: The loop back test repeats two times 2'b10: The loop back test repeats four times 2'b11: The loop back test repeats eight times Reset value 2'b01
10	MatchStatus	RO	RW	1'b1: Means the data comparison(s) between the transmitted and received data is(are) the same. 1'b0: Means the data comparison between the transmitted and received data is different. Reset value is 1'b1.
[14:11]	MismatchPosition	RO	RW	Each bit in Mismatch[3:0] indicates which flit is mismatched, don't care if the above bit is 1'b1 (matched) Reset value is 4'b0000. 4'b0101 implies flits 0 and 2 are different between what is sent and received. This field is cleared every time a loopback packet is sent. And it records the LAST mismatch if a repetitive loopback mode is configured.
15	PacketReceived	RO	RW	At least one packet was received.
[31:16]	LoopCycleCounter	RO	RW	The counter that records how many GIU clock cycles to send and receive the data. Every time a test packet is generated, this counter will be reset and keep incrementing until an expected data is received. It records the last loopback cycle count if it is set as repetitive mode.

1491

TABLE 11-1 GIULCSTR: GIU LOOPBACK CONTROL AND STATUS REGISTER ADDRESS OFFSET : 0X48

Bits	Field Name		Access	Descriptions
31:0	LoopStartDataValue	RW	RO	1. 32-bit wide starting data for loopback, it is the lowest 32-bits of the first flit. Then the rest segments are put on top of it with the same 32-bit data repeated as many times as it needs to complete a flit either in 32-byte or 64-byte long. If the loopback is configured as in repetitive mode, the lowest 32-bit of the first flit of the 2nd packet will be incremented by one, and it keeps incrementing by one for next packets included in a repetitive mode testing. 2. The lowest 32-bit of the 2nd flit carries a value of the seed data shifted to the left with number specified in field [3:2] specified in Table 11-1. Then the rest segments are put on top of it with the same 32-bit data repeated as many times as it needs to complete a flit either in 32-byte or 64-byte long. 3. Step 2 is repeated till all the flits are generated to complete a packet as shown in Table 11-3. Reset value 32'b0

1494

TABLE 11-2 THE LOOPBACK DATA SEED REGISTER (GIULDSR) ADDRESS OFFSET : 0X4C

1497

Packet No.	Flit No.	Flit Value (64B)
1	1	16*(32'h12345678)
	2	16*(32'h23456781)
	3	16*(32'h34567812)
	4	16*(32'h45678123)
2	1	16*(32'h12345679)

Packet No.	Flit No.	Flit Value (64B)
	2	16*(32'h23456791)
	3	16*(32'h34567912)
	4	16*(32'h45679123)

TABLE 11-3 AN EXAMPLE OF PACKET BASED ON A 64-BYTE FLIT WITH A STARTING DATA OF 32'H12345678, AND A LEFT SHIFT OF 4 BITS

Bits	Field Name	SW Access	HW Access	Descriptions
31:0	FirstMismatchValue	RO	RW	The first mismatched 32-bit segment, starting from LSB of a flit. Mismatched flit, each register is an 8B slice of a 64B flit where GIULDMSR_i has bits [i+31:i], i in [0,15]

TABLE 11-4 THE LOOPBACK DATA MISMATCH SEGMENT REGISTER 0 TO 15 (GIULDMSR) ADDRESS OFFSET : 0X50 + I*0X04 $i \in [0; 15]$

11.1.4 Activating loopback

To switch from functional to loopback mode the following rules must be followed :

1) Stop any functional traffic to the GIU
2) Make sure the GIU is empty by polling the TransActiv register
3) Deactivate the CXS link and wait for the link to be down.
4) Update the value of GIULDSR
5) Set the following field in GIULCSTR according to setup the loop back properties: NearLoopEn, FarLoopEn, RepeatMode, ShiftStart, DelayCounter,NumberOfTest.
6) Activate the CXS link required for proper operation
7) Write a 1 to LoopBackEn if generating traffic is requried

To switch from loopback to functional mode the following rules must be followed :

1) Deactivate the CXS link in use for loopback
2) Write a 0 to LoopBackEn.
3) Activate the CXS link for normal operation

4) When the link are in teha ctivate state, normal operation can resume.

1524

12 Acronyms and keyword definitions

12.1 CSR Register definitions

RW/RO:: Indicates this particular field or register can be only read and written by the boot-up CPU. It is read-only to other CPUs/software.

Another way to enable the above access is by using privileged access. In such a scenario, a boot-up CPU can be assigned to be in a privileged level versus the rest of CPUs are in the normal level. Currently, only APB4 protocol [2], ACE or AXI [3] protocol have the explicit control signals to enable such kind of access. However, CHI protocol [4] doesn't have a direct way to convey such a message. The limitation on CAIU for such access can be worked around it by designating a CPU that is connected to either IOAIU or APB debug port to be served as the boot-up CPU. This approach is a better way to move forward but it implies a big design change to the overall Ncore CSR handling. Needs be further examined by the design and verification teams before it can be supported.

RW: indicates this particular field or register can be read and written by any CPU/software.

RO:: Indicates this particular field or register is configured by Maestro. It is implemented as a hardware tie-off either inside of an Ncore unit or at an Ncore boundary. It is read-only to ALL CPUs/software.

12.2 Acronym definition

13 Summary of the Changes to Ncore

13.1 Global routing table.

All Ncore units will need to implement the new global routing tables.

13.2 Update to the Address map.

All Ncore units which contain an address map need to be updated with the new unit type.

13.3 Snoop filter assignment.

DCE needs to be updated to use the new mechanism to map a snoop filter entry to a cache in the system.

13.4 DVE modification

New global functionality for system messages (coherent/dvm attach sequences, events).

13.5 Legato data width and clock adaptation

Need to very 256<->512, 128<->512, 64 <->512 width adaptation.

Clock adaptation to 2GHz.

13.6 CXS gaskets (future release or if time allows)

Design AND verification of CXS asynchronous gasket.

14 Bibliography

[1] R. Holsmark, S. Kumar, M. Palesi and A. Mejia, "HiRA: A Methodology for Deadlock Free Routing in Hierarchical Networks on Chip," *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip,* pp. 2-11, 2009.

[2] ARM, AMBA APB Protocol Specification, vol. ARM IHI 0024D (ID041221), ARM, 2021.

[3] ARM, AMBA AXI and ACE Protocol Specification, vol. ARM IHI 0022H (ID040120), 2020.

[4] ARM, AMBA 5 CHI Architecture Specification, vol. ARM IHI 0050E.c (ID092622), ARM, 2022.

[5] ARM, "AMBA CHI Chip-to-Chip (C2C) Architecture Specification," vol. IHI0098, no. A, Feburary 2024.

[6] ARM, "ARM Corelink GIC-700 Generic Interrupt Controller," vol. Document ID: 101516_0400_12_en, no. 0400-12, 2024.

[7] Synopsys, Synopsys PHY IP Universal Chiplet Interconnect Express (UCIe) Advanced Package PHY Utility Block (PUB), Vols. PUB Version 1.30b, October 2, 2024, Synopsys, 2024.

[8] Synopsys, Synopsys PHY IP Universal Chiplet Interconnect Express (UCIe) Standard Package PHY Utility Book (PUB), Vols. PUB Version 1.10b, June 19, 2024, 2024.

[9] ARM, ARM DynamIQ Shared Unit-120 Technical Reference Manual, vol. Document ID: 102547_0100_04_en, 2023.

[10] ARM, ARM DynamIQ Shared Unit-110, vol. Document ID: 101381_0400_11_en, 2022.

[11] ARM, ARM Neoverse CMN-700 Coherent Mesh Network Technical Reference Manual, Vols. Document ID: 102308_0302_07_en, Issue:07, 2023.

[12] Synopsys, "Synopsys Controller IP UCIe Controller Databook," no. Version 2.41a-lca01, December 2024.

15 Appendix: ARM email exchange regarding DVM operations

Below is a record of the email exchanged with ARM about the DVM ordering requirements. The contact was Christopher Tory.

Question 1 :
Hi Hao,

> The following inquiry is based on AMBA CHI Chip-to-Chip (C2C) Architecture Specification
> version A
> ===
> In section B6.2, it introduces a definition of the DVM node. Basically, a DVM node is
> responsible for sending the local DVM request out to other DVM node(s) while collecting
> the responses for the DVM request it receives from other DVM node(s). It gives me an
> impression that ALL DVM nodes work in a peer-to-peer mode where a CENTRAL point of
> command serialization is NOT required.
>
> The below summarizes my observation on DVM related transactions and I would like to hear
> your suggestions and recommendations on the same.
>
> 1. All DVM non-sync commands are for INVALIDATIONs of TLB entry(ies), branch predicator or
> instruction cache (either physical or virtual). Thus, even though different CPUs at
> different chip managed by different DVM nodes see different order of invalidations, I
> still cannot see this behavior breaks the correctness of the operation.
>
> 2. Secondly for DVM Sync command, this is where the correct order of execution matters
> sometimes if a dependency is required. However, serializes the DVM syncs from multiple
> DVM agents is only to enforce the order the local DVM node can observe even in a
> standalone configuration. Due to the various delays between different DVM agents and the
> local DVM node, the order the local DVM node can maintain still might not the right
> order multiple DVM Sync commands wanted. Since the correct order of such can only be
> guaranteed by using a semaphore(s) between and among them.
>
> 3. This scenario can be easily extended to a multiple chip scenario where I think
> serialize the DVM commands across multiple chip at a central location is not necessary.

Your understanding here appears correct.

Each DVMOp / DVMSync ordering sequence from a specific RN is independent from DVMs issued from a different RN.

Each RN can issue many DVMOps which will get completed by the other RNs in any order. Once the issuing RN has received the Comp responses for all the DVMOps that it requires to be observable, it then issues a DVMSync. When this completes, it is guaranteed that the effect of all the required DVMOps have been completed that were issued from this RN and will be observable.

If multiple RNs are issuing DVMOps, and some synchronization between them is required, then you would need some kind of software based synchronisation as you described.

As long as a specific RN always sends its DVMs to the same DVM Node, then I would concur that DVM Nodes are effectively peer to peer and there is no centralized ordering between them.

Regards,
Chris

Question 2 :

Hi Hao,

> Thanks for the reply, we have one more clarification as the following:
>
> ARM CHI version G. IHI0050G, section B4.11. It says: "In addition to many Requesters
> issuing transactions at the same time, the protocol also permits each Requester to make
> multiple outstanding requests, and to receive multiple outstanding snoop requests. The
> interconnect, that is, ICN(HN-F, HN-I and MN), is responsible to ensure that there is a
> defined order in which transactions to the same cache line can occur, and that the
> defined order is the same for all components."
>
> My understanding is that MN is the node handles DVM related transactions. With that, can
> you please further clarify if the above invalidates the conclusion earlier?
>
> I think our understanding still holds since the defined order isn't required for the same
> cache line just similar to the scenario of the DVM sync operations discussed earlier.

Yes, your understanding is correct.

This also relates to how DVMs work - the MN will issue these to the RNs as SnpDVMOps, but there's no indication back to the MN about whether these are completed until a DVMSync is used. The RNs can receive many DVMOps and execute these in any order, and the Comp response just indicates that the DVMOp has been received (and would be ordered against a future DVMSync) - in other words, the MN cannot ensure that the DVMOps occur in a defined order until a DVMSync is used.

By contrast, the section you reference is about the data paths. In this case, mechanisms like CompAck do indicate to the HN that a request has completed at the RN, and so the HN can ensure these happen in a defined order for each transaction.

Regards,
Chris

Question 3:
Hi Hao,

> If our understandings are correct, MN should be taken out of the list of "ICN (HN-F, HN-I
> and MN)" for the paragraph referred in the CHI.G specification, right?

Yes, or that section should be re-written slightly.

The MN doesn't affect the ordering of transactions to the same cache line. The MN does do some ordering for DVMOps if it gives an early Comp response - it must ensure that the DVMSync gets correctly ordered against the DVMOps that it has sent the Comp for in this case.

I'll file something internally to have this clarified.

Regards,
Chris