Arm Chiplet System Architecture

Document number	DEN0145
Document quality	00bet0
Document version	A
Document confidentiality	Non-confidential
Document build information	c1288af

Arm Chiplet System Architecture

Release information

Date	Version	Changes
2025/Jan/22	A	• Beta release.
2025/Jan/20		• Clarified system memory in I/O and I/O controller chiplet types.
2025/Jan/10		• Refined and updated rules on trace interface and system counter (for trace) for all chiplet types.
2025/Jan/10		• Added requirements for debug to the "Remote Translation" chiplet types.
2025/Jan/10		• Added detail to chiplet identification register rules.
2024/Nov/21		• Refined and updated rules on trust boundaries & HES for chiplet types.
2024/Nov/11		• Clarified use of system main memory. Corrected rules on MMU placement.
2024/Oct/15		• Added section on Hardware Faults in Chiplet Interfaces.
2024/Oct/08		• Alpha 2 draft.
2024/Sep/27		• Annotated rules with compliance levels.
2024/Sep/16		• Clarified rules around primary Hub and Compute 2 chiplets.
2024/Aug/21		• Added rules requiring ITS for hosting chiplet types.
2024/Aug/13		• Added system rules for GIC affinity routing, CPU MPIDR.
2024/Aug/01		• Added Trace interface to some chiplet types.
2024/Jul/31		• Added Debug Access interface to some chiplet types.
2024/Jul/04		• Alpha 1 draft.
2024/Jun/26		• Added the Fully Coherent Expansion (Remote Translation) Chiplet.
2024/Jun/24		• Added support for the interfacing of Compute 2 and Hub chiplet types.
2024/Jun/07		• Updated GIC Requirements.
2024/May/22		• Added Non-coherent memory interface to chiplet interface definitions.
2024/May/14		• Updated debug cross-trigger interface implementation.
2024/Apr/18		• Modified "system counter" component rules to include synchronization chaining.
2024/Apr/15		• Added new common component: "system control agent".
2024/Apr/08		• Added support for tiled Compute 1 and system expansion chiplets.
2024/Apr/04		• Added fully coherent, I/O coherent and non-coherent devices to chiplet type rules.
2024/Apr/03		• Added three chiplet types: Fully Coherent System Expansion Chiplet, I/O Chiplet, I/O Controller Chiplet.
2024/Apr/03		• Added optional system memory to C1 Chiplet.
2024/Mar/18		• Broadened rules about expansion chiplet MMU placement to include all agents.
2024/Mar/01		• Separated "Debug" and "Trace" rules.
2024/Feb/12		• New Threat Model and Security Goals chapter.
2024/Jan/29		• Clarifications to "System counter" component.
2024/Jan/15		• Aligned wording and ordering of rules common between chiplet types.
2024/Jan/05		• Combined interface descriptions for I/O Coherent Chiplet variations for Hub and Compute 2 chiplets.
2023/Dec/15	A	• Alpha 0 draft.

Arm Non-Confidential Document License (“License”)

This License is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this License (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this License. By using or copying the Document you indicate that you agree to be bound by the terms of this License.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“Licensee”) is subject to the terms of this License between you and Arm.

Subject to the terms and conditions of this License, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide License to:

(i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;

(ii) manufacture and have manufactured products which have been created under the License granted in (i) above; and

(iii) sell, supply and distribute products which have been created under the License granted in (i) above.

Licensee hereby agrees that the Licenses granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm’s view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

Reference by Arm to any third party’s products or services within this document is not an express or implied approval or endorsement of the use thereof.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENSE, TO THE FULLEST EXTENT PERMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENSE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE’S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENSE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This License shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this License then Arm may terminate this License immediately upon giving written notice to Licensee. Licensee may terminate this License at any time. Upon termination of this License by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this License, all terms shall survive except for the License grants.

Any breach of this License by a Subsidiary shall entitle Arm to terminate this License as if you were the party in breach. Any termination of this License shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

The Document consists solely of commercial items. Licensee shall be responsible for ensuring that any use, duplication or disclosure of the Document complies fully with any relevant export laws and regulations to assure that the Document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

This License may be translated into other languages for convenience, and Licensee agrees that if there is any conflict between the English version of this License and any translation, the terms of the English version of this License shall prevail.

The validity, construction and performance of this License shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Arm document reference: PRE-21585

version 5.0, March 2024

Note on the Beta Release

In accordance with the proprietary notice contained within this document Arm grants implementation rights to this system architecture specification, allowing you to use it to build a product that meets the requirements defined within this specification. Beta specifications are subject to change. Kindly inform us if you are utilizing this specification so that we can keep you informed of changes. Please register your interest with us by sending an e-mail to csa-feedback@arm.com.

If you reference the Arm Chiplet System Architecture in the marketing materials for your product, Arm expects you to have conducted reasonable due diligence to verify that your product follows the CSA requirements. Additionally, you should specify the CSA chiplet type that you are claiming alignment with, and the CSA defined interfaces that are supported.

Arm provides a checklist to help designers check that their design follows the CSA requirements. We will supply the checklist to anybody constructing a CSA compliant solution and will review completed checklists that are returned to Arm.

Contents

Arm Chiplet System Architecture

Conventions

Typographical conventions

The typographical conventions are:

italic

Introduces special terminology, and denotes citations.

bold

Denotes signal names, and is used for terms in descriptive lists, where appropriate.

`monospace`

Used for assembler syntax descriptions, pseudocode, and source code examples.

Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.

SMALL CAPITALS

Used for some common terms such as IMPLEMENTATION DEFINED.

Used for a few terms that have specific technical meanings, and are included in the Glossary.

Red text

Indicates an open issue.

Blue text

Indicates a link. This can be

- A cross-reference to another location within the document
- A URL, for example http://developer.arm.com

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by `0b`, and hexadecimal numbers by `0x`. In both cases, the prefix and the associated value are written in a monospace font, for example `0xFFFF0000`. To improve readability, long numbers can be written with an underscore separator between every four characters, for example `0xFFFF_0000_0000_0000`. Ignore any underscores when interpreting the value of a number.

Pseudocode descriptions

This book uses a form of pseudocode to provide precise descriptions of the specified functionality. This pseudocode is written in a monospace font. The pseudocode language is described in the Arm Architecture Reference Manual.

Assembler syntax descriptions

This book contains numerous syntax descriptions for assembler instructions and for components of assembler instructions. These are shown in a `monospace` font.

Rules-based writing

This specification consists of a set of individual *content items*. A content item is classified as one of the following:

- Declaration
- Rule
- Goal
- Information
- Rationale
- Implementation note
- Software usage

Declarations and Rules are normative statements. An implementation that is compliant with this specification must conform to all Declarations and Rules in this specification that apply to that implementation.

Declarations and Rules must not be read in isolation. Where a particular feature is specified by multiple Declarations and Rules, these are generally grouped into sections and subsections that provide context. Where appropriate, these sections begin with a short introduction.

Arm strongly recommends that implementers read *all* chapters and sections of this document to ensure that an implementation is compliant.

Content items other than Declarations and Rules are informative statements. These are provided as an aid to understanding this specification.

Content item identifiers

A content item may have an associated identifier which is unique among content items in this specification.

After this specification reaches beta status, a given content item has the same identifier across subsequent versions of the specification.

Content item rendering

In this document, a content item is rendered with a token of the following format in the left margin: L_{iiiii}

- *L* is a label that indicates the content class of the content item.
- *iiiii* is the identifier of the content item.

Content item classes

Declaration

A Declaration is a statement that does one or more of the following:

- Introduces a concept
- Introduces a term
- Describes the structure of data
- Describes the encoding of data

A Declaration does not describe behaviour.

A Declaration is rendered with the label *D*.

Rule

A Rule is a statement that describes the behaviour of a compliant implementation.

A Rule explains what happens in a particular situation.

A Rule does not define concepts or terminology.

A Rule is rendered with the label *R*.

Goal

A Goal is a statement about the purpose of a set of rules.

A Goal explains why a particular feature has been included in the specification.

A Goal is comparable to a “business requirement” or an “emergent property.”

A Goal is intended to be upheld by the logical conjunction of a set of rules.

A Goal is rendered with the label *G*.

Information

An Information statement provides information and guidance as an aid to understanding the specification.

An Information statement is rendered with the label *I*.

Rationale

A Rationale statement explains why the specification was specified in the way it was.

A Rationale statement is rendered with the label *X*.

Implementation note

An Implementation note provides guidance on implementation of the specification.

An Implementation note is rendered with the label *U*.

Software usage

A Software usage statement provides guidance on how software can make use of the features defined by the specification.

A Software usage statement is rendered with the label *S*.

Additional reading

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *Arm Base System Architecture*. (ARM DEN 0094) Arm.

[2] *Arm Architecture Reference Manual*. (ARM DDI 0487) Arm.

[3] *AMBA 5 CHI Architecture Specification*. (ARM IHI 0050) Arm.

[4] *Arm Generic Interrupt Controller Architecture Specification: GIC architecture version 3 and version 4*. (ARM IHI 0069) Arm.

[5] *Arm CCA Security Model version A.d*. (ARM DEN 0096) Arm.

[6] *Arm CoreLink GIC-700 Generic Interrupt Controller Technical Reference Manual*. (ARM 101516_0300_10_en) Arm.

[7] *Arm CoreSight Architecture Specification, v3.0*. (ARM IHI 0029F) Arm.

[8] *Arm Embedded Trace Router Architecture Specification*. (ARM IHI 0081A) Arm.

[9] *Arm System Memory Management Unit Architecture Specification: SMMU architecture version 3*. (ARM IHI 0070) Arm.

[10] *Arm Realm Management Extension (RME) System Architecture*. (ARM DEN 0129) Arm.

[11] *Arm Power Policy Unit Architecture Specification*. (ARM DEN 0051) Arm.

[12] *Universal Chiplet Interconnect Express (UCIe) Specification Revision 1.1*. UCIe.

[13] *MIPI Alliance Specification for I3C*. MIPI.

[14] *Arm AMBA CHI Chip-to-Chip Specification*. (ARM AES 0071) Arm.

[15] *PCI Express Base Specification*. PCI-SIG.

[16] *Arm CoreLink GIC-600 Generic Interrupt Controller Technical Reference Manual*. (ARM 100336_0106_00_en) Arm.

[17] *Arm System Control and Management Interface Platform Design Document*. (ARM DEN 0056) Arm.

[18] *Arm Product Security*. See https://www.arm.com/products/product-security

[19] *AMBA ATB Protocol Specification*. (ARM IHI 0032Cs) Arm.

[20] *AMBA AXI Protocol Specification*. (ARM IHI 0022) Arm.

[21] *Compute Express Link (CXL) Specification*. CXL.

[22] *Arm Architecture Reference Manual Supplement: Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture*. (ARM DDI 0598) Arm.

[23] *Arm Power Control System Architecture*. (ARM DEN 0050) Arm.

[24] *IEEE Standard Test Access Port and Boundary-Scan Architecture*. (ISBN 0-7381-2945-3) IEEE.

Feedback

Arm welcomes feedback on its documentation.

Feedback on this book

If you have comments on the content of this book, send an e-mail to csa-feedback@arm.com. Give:

- The title (Arm Chiplet System Architecture).
- The number (DEN0145 A).
- The page numbers to which your comments apply.
- The rule identifiers to which your comments apply, if applicable.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests PDFs only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the appearance or behavior of any document when viewed with any other PDF reader.

Chapter 1
Introduction

Systems are being produced using chiplets instead of traditional monolithic dies for various reasons, including removing reticle limit restrictions, yield improvements, reuse, enabling multiple product configurations, and optimizing technology cost and applicability.

Chiplets are being used to partition monolithic designs, but also to aggregate additional functionality, such as devices traditionally in separate packages, into a shared package. The Chiplet System Architecture is applicable for a monolithic system design partitioned across chiplets, where chiplet interfaces replace on-die connections. For partitioned systems, the CSA provides a system architecture defining common partitioning approaches. When building chiplets to aggregate peripheral devices connected through existing standards, for example PCIe, into the same package then mature system architecture exists, hence there is no additional value that can be added by the CSA. Arm believes both approaches are equally important for building chiplet-based systems.

This Chiplet System Architecture (CSA) document specifies a hardware system architecture for an Arm system that is distributed over multiple chiplets. An *Arm system* is defined as the hardware that supports a single system software image built for the Arm 64-bit architecture, such as an operating system.

The CSA identifies a set of high-level properties for a number of system topologies and their constituent chiplet types. It specifies the chiplet types required to build these systems, their properties, and their interfaces. Separate from the chiplet definitions it specifies protocol and transport mechanisms for each interface requirement.

1.1 Purpose

This specification is intended to act as a foundation for an Arm-based chiplet ecosystem built around the reuse of components and design effort in chiplet based systems, up to and including the reuse of complete chiplets.

For this ecosystem to succeed there needs to be agreement upon a classification of the different types of chiplets, so that the features and interfaces are compatible. This classification forms the basis for additional layers of standardization such as protocols, physical interface definitions and reusable chiplet products.

Throughout the design process of a chiplet, implementation specific choices are made about how to realize system functionality, for example, interrupts or DMA. Where these choices are arbitrary and non-differentiating, the CSA provides a consistent implementation. This enables chiplet compatibility without limiting differentiation.

1.1.1 Document Structure

The CSA specifies system compositions of chiplets, the chiplet types and their interface definitions.

Figure 1.1 shows the relationship between the components specified in the CSA.

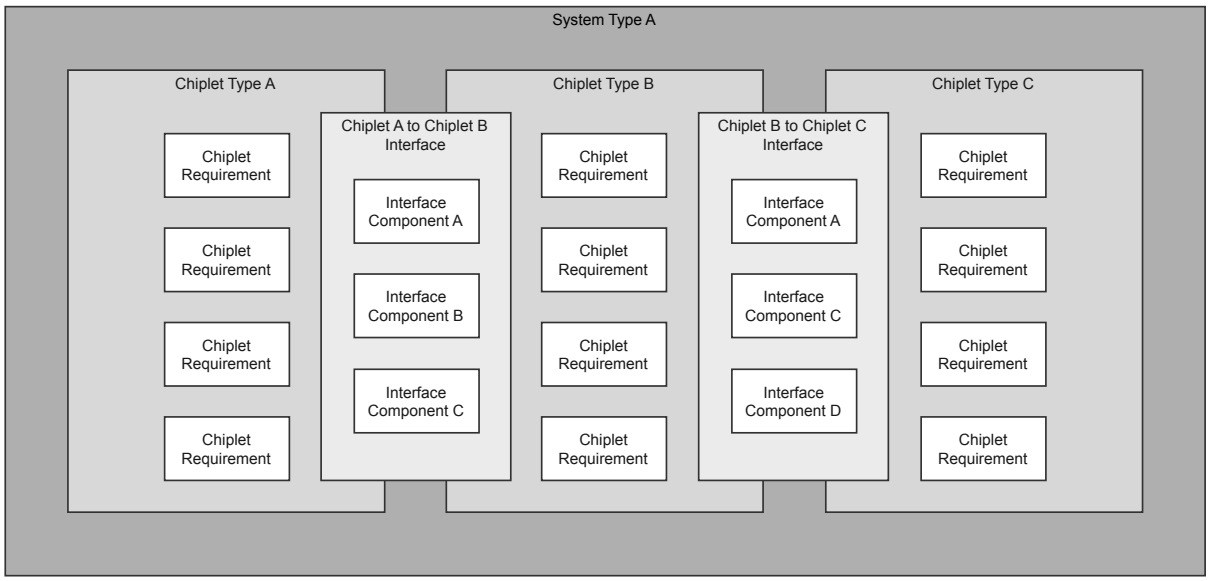

Figure 1.1: CSA component relationship

A system is distributed across multiple chiplets, each chiplet is of a defined type. The CSA specifies a valid composition of chiplet types for that system.

A chiplet type specifies the properties of the chiplet that fulfil its requirements within the system. These usually have an effect on its interface requirements to other chiplets in the system.

A chiplet interface specifies the properties of the connections between two chiplets of the same or a different type. A chiplet interface is composed of multiple interface components, each describing the properties of an interface requirement.

Where possible, a chiplet interface component is mapped to an implementation that specifies a protocol and transport mechanism, along with any additional requirements. The CSA specifies protocols and transport mechanisms where appropriate Arm or industry standards exist, otherwise the CSA maps the interface component as IMPLEMENTATION DEFINED. The CSA does not specify any new interface functionality.

1.2 Using this specification

The use of this specification depends upon the perspective of the user. This could be, among others, a system designer looking to partition or make a system using chiplets, or a standalone chiplet designer, either within the same organization or a third party designing a compliant chiplet.

Whilst the scope of the CSA does not cover the entirety of the solution space for connecting chiplets, it covers an important part of the functional interaction and interfacing between chiplets.

1.2.1 System Designer Considerations

A system designer is building a system with chiplets, typically, where some are reusable components either over product configurations or multiple product generations. Some chiplets they use might be sourced from standalone chiplet designers.

For a system designer:

- Identify the system type that best matches the requirements. Some considerations could be:
  - Chiplet partitioning alignment with system requirements
  - Product configuration requirements provided by chiplet combinations, such as:
    - Products with a different scale, for instance, different amounts of compute capacity
    - Products with different additional functionality, such as different acceleration engines
- Review the requirements of the component chiplets to identify the number and types of chiplets that are the best match for partitioned functionality. This includes mandatory chiplets to make a minimally functional system and either additional mandatory types, potentially for scaling reasons, or optional types to add additional differentiating functionality. Some considerations could be:
  - Chiplet size, for reticle/yield implications
  - Technology cost / applicability
  - Re-use potential, either from past designs or for this design in the future
  - Specialist functionality separation
  - Availability of 3rd party chiplets, either on the market or as a design service

These along with other considered factors, could be reasons for the selection of the number, and optional types, of chiplets within the system.

Then the selected chiplet types can be used as a base specification for the system and chiplet design, or to reference in an RFQ for a third party chiplet designer.

1.2.2 Standalone Chiplet Designer Considerations

A standalone chiplet designer is building a chiplet that is to be used in other systems. This could be for a specific system designer or intended to be applicable to multiple system integrations.

For a chiplet designer:

- Use the system types to understand how the intended chiplet functionality integrates with different systems.
- Understand the chiplet requirements, either from internal or external (RFQ) requirements.
- Select the appropriate chiplet type, by capabilities and interfacing requirements, or from external (RFQ) requirements.

Then the selected chiplet type can be used as a base specification for the chiplet design.

This ensures that the standalone chiplet design integrates the correct functionality and interfaces to connect to a system that also follows the CSA.

1.2.3 Chiplet types and classification

A CSA chiplet type is defined by a set of requirements that it must meet, these are both a set of on-chiplet functionality and a mandatory, but not limiting, set of interfaces. These requirements ensure that when the chiplet is included with other chiplets in the system type they create a functional system.

A chiplet with additional capabilities can still be classified as a chiplet type with less functionality if it contains the correct functions. This could lead to a Chiplet being able to be classified as more than one type. This might need integration specific configuration of the interfaces if certain functions must be managed to ensure correct behavior as the specified chiplet type.

Having a chiplet support multiple types could make a chiplet more flexible, allowing it to fit into more than one system. However, it could be suboptimal, as functionality included to make it another chiplet type is unused, potentially wasting area and power.

1.2.4 Interface types and classification

A CSA chiplet interface type is defined by a set of requirements that it must meet, these requirements are defined by the chiplet types that are being connected and ensure the defined chiplet types can interact, so they create a functional system.

An interface might be able to interact with more than once chiplet type if it contains all the required functions in the interface and in the chiplet. This could lead to an interface being able to be classified as more than one type. This might need configuration of the interface at manufacturing or runtime to support connecting to a particular chiplet type.

Having an interface support more than one type allows flexibility in the use of an interface, and therefore the chiplet. However, it could be suboptimal, as functionality included to make it another interface type is unused, potentially wasting area and power.

1.3 Relationship to other standards

The scope of the CSA is the hardware that is visible to the system software, this is the same scope as the Arm Base System Architecture (BSA) [1]. However, there is no direct dependency between the CSA and the compliance of the system with the BSA. The CSA can apply to systems that do not support the BSA. The CSA describes the classification of chiplets, whereas the BSA describes the functionality required within the system as a whole for the purpose of consistent software deployment. The two specifications are complementary and can be used together to build the required system using chiplets.

The CSA includes a classification of Arm-based chiplets, including interface requirements. For example, two interconnected chiplet types that both contain Arm PEs require a suitable protocol to exchange coherent memory transactions and supporting messages. The CSA defines these interface requirements and maps them to existing industry standard protocol and transport mechanisms in Chiplet Interface Implementation. The CSA does not contain any specifications for protocols or standards, it relies on existing standards, although it might indicate a requirement for a standard where one does not currently exist.

While not directly in the scope of chiplets described in the CSA, PCIe and CXL device chiplets and chips can still, and are expected to be, attached to a CSA compliant system.

1.4 Limitations of scope

Hardware that does not relate to a single Arm-based system is outside the scope of CSA. These functions are still expected to be used by an Arm-based system that complies with the CSA, but the CSA does not specify any rules that apply to them. Examples of this type of hardware include:

- Aggregation of off-die devices and accelerators that are discoverable and abstracted by a device driver model, such as an industry standards-based peripheral device.
- Integration between separate Arm-based systems, as in the case of a server compute chiplet and DPU chiplet where each is a separate Arm-based system. The CSA can be applied to these systems separately.
- A chiplet that is wholly abstracted from the Arm system, such as a JEDEC standard memory chiplet.

1.5 Basic Definitions

The following terms or concepts are used throughout the CSA.

Arm System or *System*

The hardware that supports a single system software image of an operating system or hypervisor built for the Arm architecture. It defines the scope of the CSA specification. “Arm System” and “System” are both used in the CSA and have this definition.

Application PE

An ARM A-Profile Processing Element (PE) that supports the system application.

NOTE: Other processing elements might exist in the system that do not run the main system software, such as those that run security or system control firmware. In the terminology of the CSA these are not considered to be Application PEs.

Fully Coherent Device

An application specific device such as an accelerator for a particular type of computation. The device incorporates coherent caches and makes coherent accesses to the rest of the system.

I/O Coherent Device

An application specific device such as an accelerator for a particular type of computation. The device supports I/O coherency, having no coherent cache of its own and making coherent accesses to the rest of the system.

I/O Coherent I/O Device

A I/O controller and interface that supports I/O coherency, having no coherent cache of its own (or in devices connected to the interface) and making coherent accesses to the rest of the system.

Non-Coherent I/O Device

A I/O controller and interface that accesses only Non-cacheable system memory.

System Addressable Memory

All agents in the system share a common physical address map that includes Normal and Device memory, referred to by the CSA as “System Addressable Memory”. The Arm ARM [2] assumes that all Application PEs that use the same operating system or hypervisor are in the same Inner Shareable shareability domain. The CSA assumes that all Application PEs in the system share the same operating system or hypervisor. All the Normal memory of the system is potentially cacheable, therefore it is Inner Shareable or Non-shareable.

System Main Memory

Memory that is both:

- contained in a chiplet, either integrated or directly connected, and
- allocatable as Normal memory by the OS or hypervisor.

Aligns with the Arm ARM [2] definition of *Conventional Memory*: “memory locations from which generic OSs and application runtimes expect to create allocations for general software use”.

Aligns with the CHI [3] definition: “the memory that holds the data value of an address location when no cached copies of that location exist”.

Typical examples include external DRAM, or HBM.

Boot Code Image, *Boot Image*

An image of firmware for one of the boot stages of a Trusted Security Agent, System Controller or an Application PE.

Chapter 2
Chiplet System Architecture Compliance Levels

IYKNYP This section describes the levels of compliance within the CSA and how they can be applied.

2.1 Introduction

The Chiplet System Architecture (CSA) embeds the notion of levels of compliance. Each level adds functionality over and above a previous level. Unless explicitly stated, all specification items that belong to level N apply to levels greater than N.

A chiplet or system can claim compliance with a level when it adheres to all the rules for that level and all lower levels.

Individual rules are marked with their levels, as in the example below:

R_{XXXXX} This is an example of a CSA rule that is assigned to level N.

(LEVEL-N)

Rules without an indicated level needs to adhered to if the chiplet or system wants to reach full compliance to the CSA.

2.1.1 Level 0

This level means the chiplet meets the functional partitioning and interface rules for the chiplet type specified in sections Chiplet Types and Chiplet Interfaces.

A list of the level 0 rule identifiers that appear in the CSA is provided in Table 2.1.

2.1.2 Level 1

In addition to the requirements in level 0, for compliance to level 1 the design meets the requirements in:

- Coherent Memory Traffic Interface.
- I/O Coherent Memory Traffic Interface.
- Non-Coherent Memory Traffic Interface.
- Device Configuration Interface.

A list of the level 1 rule identifiers that appear in the CSA is provided in Table 2.2.

2.1.3 Full Compliance

This indicates full alignment to the CSA requirements for the specified chiplet or system.

A list of the level Full rule identifiers that appear in the CSA is provided in Table 2.3.

2.1.4 Rule Identifiers by Level

Table 2.1: Level 0 Rule Identifiers in the CSA

3 System Types							
3.1 Compute and Hub System							
LYHKS	TFJSM	LCJNF	CJZHL	ZCFRY	PPLNQ	MDPNF	FZVCL
DXYWQ	GBSPR	JLWBK	ZKZHH	DVHYW	QXKKG	FTXSZ	PZWRN
HLQVQ	TYGPR						
3.2 Compute Tile System							
TKPTF	MVWBD	JHWVM	DSKBW	ZFTDV	BFSJV	ZDVGZ	CCHPT

WJJVL	KSQJG	WTRKM	VNDCV	NLRPY	SCNYP	SHYRP	LBCYZ
SMCPF							

4 Chiplet Types

4.1 Compute 1 Chiplet

BPRJK	RCYDP	JWRCG	XTDHG	PTZSC	NCHRQ	KNKSV	NRJZH
NGZDK	KRCKM	XGXZR	CQDKD	GHRFJ	CKTDK	JBYBY	HDRKM
KKYSW	MJNJG	GVVMS	DZFZV	ZDSHM	FTKXX	RNSKY	PGGRB
XZXQB	NXBYL	MGWMP	XSFHF	SFPHM	ZMQTH	CMQPJ	MDGWP
QXJHT	LWGXS	KMBXK	MRRGT				

4.2 Hub Chiplet

MZLJX	CNQZH	LCQGC	SVWVP	HBYND	MKYNH	WMTVP	MBQZT
SPRHJ	MLQLD	ZTCKN	PCJMB	FXFNL	JDYJY	PMKFN	YHQDR
YVFHS	RXLZG	MTBPW	VJFWY	KZFYG	FQRZT	JNXMC	MSJZQ
MPFHT	NRZBH	PXHHJ	FZDYK	FTXZM	ZPLGW	VNGDP	SDDYH
XLLCC	RFPQN	NJQDV	NXDVW	CKXYX	ZFWWC	PYRRZ	QCBRF
PLXZG	XSBBL	ZCPHB	HCCDW	LDSVP	GYSJB	BJBSF	QPDWG
LGWZM	SXNDJ	TTRSK	GNBZX	GHVLC	YJBND	KYYNQ	DZGQK

4.3 Compute 2 Chiplet

SJNGN	NDHDT	KNHMM	CGKQW	MDFWD	WLHVZ	JZSRC	DVNHD
QGGWQ	LGXLN	XNYDJ	PHHQM	ZVZZN	VYJFL	ZCLCN	FXTRR
XCVGN	ZXWKH	DZDHQ	PBYQY	BBBJT	RWFFF	KYSDY	VGCBH
MRYYB	XBBLT	LYFPT	THPPG	NYZVM	YNMLX	SGZXK	VVDFD
SCYSD	VVNTG	NLGDR	WVMSK	PGTMQ	YQMHG	FDRRV	MSCJJ
RXMNK	LGHFC	FWCJG	FZDJN	KGFNL	GNZNV	BGJCM	QHPNB
VRVYH	ZDBLR	CJQRD	CKWDT	DPXXH	XDDFR	XYVWT	LPTTQ

4.4 Fully Coherent Expansion Chiplet

SHXTT	MKCTS	HMZJQ	YDWVV	NBXWR	FMBVG	JNCKH	VCPQR
FJDNP	SPTFW	FQPML	QDMPX	LYVBX	DWFSK	XVWQF	QQTXS
RDGJV	LDYRT	CXLDV	DDBRW	NPQTP	FPNKN	CSCZG	MCDLS
ZRPHF	QZSYG	PWHNJ	MBCXY	QCHVW	FVWGK	VRPMH	ZNXXR
RJMNL	CVMFD	PDTQR	ZKPSD	FFWWY	BMPMM	FCCGZ	HCQGD
ZDSMN	CMTXV	ZJLZD	BDDYL	VJGGP			

4.5 Fully Coherent Expansion (Remote Translation) Chiplet

KGJQZ	KSDBW	MHJBP	GGJVW	WDGDV	KLQNT	TFZGY	VHJYG
SVMSC	FFDXH	CDQVL	HCGBH	KYJPH	KTTXB	BJTHM	PJVLK
JBRRP	PDQFV	LMJFS	XTTWV	MBSCP	XVZTS	PYTWQ	JCTLV
JSDXW	WXYFJ	LNNVR	MKBNF	XBJSZ			

4.6 I/O Coherent Expansion (Translated) Chiplet

YTTPM	RWJXM	NJQMS	FHVQW	XPQFB	DFJJB	LRYYF	NKBDT
GPNPF	JKNBM	YLRSX	TXQKB	MNCCC	XCBFQ	BGYTR	YSCVM
XDDMZ	XWPCG	DGKNW	VCSNX	HMGLJ	ZHDSR	LPFHQ	SXYNL
MPXKH	TKMWG	MGQGR	YXVZS	TWRHN	ZHTQD	BVSBW	PRMCJ
XJDKV	BXZMH	KKYCW	LCWDK	JTTCQ	QLSDY	TWVMQ	ZKGSN
JYTSS	KKPCM	XPJTN	DDQVP				

4.7 I/O Coherent Expansion (Untranslated) Chiplet

YWWFY	WFLXB	JHQPD	BNCHJ	DGKSX	HKCGS	YHDSL	DPLFM
NDGMQ	RGMBN	NPHFD	KFPMT	VSTXF	KWSDM	WFVTN	GXRYN
YWQVN	YHQVD	PJYKG	KMTCS	JMJYS	KXJWP	TGDQV	BVFLF
LDHBH	HHJCC	MGPTM	YQFDX	LXWHF	PJLCZ	DXFGF	JZJYR
WSWRW							

4.8 I/O Coherent Expansion (Remote Translation) Chiplet

MCTLV	DFKTT	XKHNG	RXVFP	SJJRK	VGYHF	SYMRV	MGPMV
MJDDS	VLFCJ	XDGPZ	TXXQS	KCVLT	FJJKL	ZNCPC	SSKPN
VPTCL	CKXDG	LCWHG	DDZZJ	VVWXF	MTJGB	GMXLY	JWTCB
WXTJK	MKFTB	PFTPX	HNKXF	DZVCK			

4.9 I/O Chiplet

FKNWS	CFTXC	FHZFP	ZQPNC	JHMFZ	KWXTX	CPTMG	ZMHCY
KHZVP	YRKBX	JSKVP	RRCYS	KXVHZ	GDTMR	KCVWN	TGMWL
JSWCC	GDKCJ	WKBWW	LCHML	XJFYN	CTTLN	TKBBC	XTWDY
WPDRB	BNDMC	HGNYQ	RCNGP	YFNFR	FWSCR	YDSWZ	BXGXT

4.10 I/O Controller Chiplet

FFNKP	RDQYR	DSBQD	WBVMW	XWTXV	BNQNG	JRSRT	RZGDN

QSWDY	LZRDM	MWPHD	DWLLK	CKFDQ	VHQZD	NJRSR	GXMYX
VSNSX	BMYSB	DTLYG	KZQVC	PDQHN	KKFWR	JQYJN	KDXCW
GRYYV	JDSWQ	HWYHJ	NVSMJ	NHPSD	PBFBD	GNKGG	VVBCX

5 Chiplet Interfaces

5.2 Hub Chiplet to Hub Chiplet Interface Requirements

ZPXZW	SPPXN	KMPRP	SYDRX	LQZMW	RXQWQ	CCNRY	DFVLC
CQVLD	SSPZM	RSSMW	DHBHJ	TPSJG	DKMDM	RVKPZ	NJPCS
PMLBM	RJNMR	VCBTT	BMRTC	YWJQM	YYTTD	RNHMS	MRYRH

5.3 Hub Chiplet to Compute Chiplet Interface Requirements

NQBSN	SPXCX	YWSTJ	FVFQQ	GHLHT	YHFBR	YPLLB	CLWWD
GMZTH	ZYSFB	DQDKT	NNQCD	FXRMW	SZZGL	YGWKV	BHDVN
HJZYS	HXDQV	JNCKG	HZCKZ	GDTQB	CJCVS	MSRQF	SPMCF
WFTMD							

5.4 Compute 1 Chiplet to Compute 1 Chiplet Interface Requirements

SZTCH	PMVRY	RSGWB	LPBNG	KYGBK	RBVXV	TKLSJ	SDKFD
JSDGJ	QBRTM	BPTLC	KFDDX	YYBCT	LPQPB	SSDSP	FZBGN
SCNVZ	ZRCQC	DSHTS	MFVBR	YMRQK	TKKSY	LSFZQ	MSMMY
PHLYM							

5.5 Compute 2 Chiplet to Compute 2 Chiplet Interface Requirements

JRSXS	MKCPL	RLQYC	RBNSS	CZLQD	SKNMG	BKKWV	NLNCD
JRLRN	DSNRH	SBJDB	TXHMX	LLTTW	GZMKP	RTCSK	PDJZG
NLZFW	JRCFK	WYDPS	QLHKH	QVHXK	HNFBT	QTVRH	RXSYB

5.6 Fully Coherent Expansion Chiplet Interface Requirements

PRKPY	VFPKB	WPZNH	VLQPR	RBMCH	NTBDH	QTGWK	WGSPV
PFZGF	XVQPJ	BYBMM	GJBYM	JZNWW	JZKYZ	RVDHX	ZCQQH
GHGXC	GQTVW	LJRBQ	YMVWW	KMLZL	FZVDD	RZHPT	YDDYT
BNFXW	JSNRX	HYMKW					

5.7 Fully Coherent Expansion (Remote Translation) Chiplet Interface Requirements

CMRXD	TGCLP	QSHDG	SXWZM	VBVKL	LHSRL	NKXXM	NBWLZ
LWJBR	HYTDS	GLVZD	LXTVM	GRXQH	CSFZB	GGTMJ	QBXFV

BMWVT	FWJVY	PCWPV	WWKGM	FVSKN	YBDFB	VSTHC	WKLYR

5.8 I/O Coherent Expansion (Translated) Chiplet Interface Requirements

CNCGH	XTJPP	JLGGN	HCYKF	HXNJH	RGQKJ	YHSGY	JSFTT
BMJZP	RNYKY	XRDCX	ZGDGY	JJKSQ	BHRHJ	NTZHB	SQLJT
CMRMD	WDLDK	CYHBV	GRCDW	ZNNHF	NTBDN	RMTPR	ZCDGQ

5.9 I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements

WBSBR	TJPJV	WWJWH	CBWZD	SZXHJ	QBQMD	DTDLR	CDBPS
RYVZX	QTJWG	BSQPC	MCCDT	PVDQS	PDJPV	CDVBW	ZGNCM
PPGVB	DJZLN	JJPDD	JSFJW	JWPNB	KSYQB	QXLWL	

5.10 I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements

HHPXJ	YQSDV	MTLNR	SVYMX	NMQCH	ZLKGN	QRJPN	RVRZS
LFDBC	ZTFJZ	BNLPG	LGJSW	VQBCT	FQQRQ	KDYYR	MGPHF
SDPSD	KVYCY	BKNRG	HQWXD	NVKKZ	KZNLW		

5.11 I/O Chiplet Interface Requirements

WGMMT	PVTQN	GBYYK	LHRDK	WLLKW	XKHXM	DVBZV	RSJDZ
DFQYV	PMXFG	YLRCM	HJDFT	RVTJM	KBGTC	HXWGL	HHJRZ
DDSPX	PTQBD	LVRMZ	JVJDH	TDQTB	XLNKX	QHJWB	QLRQL
SRTTF							

5.12 I/O Controller Chiplet Interface Requirements

RDVNB	CDZVL	JHJVP	DDDSW	YZHWP	YJGHH	LSJMB	PCWSQ
ZQVFY	YRHHX	HLXNC	LKCWK	XZPFB	SXXYP	JSHJH	WBVWT
SBNMJ	RNDMS	TQCXK	QQNKF	JRYTT	NPCXN		

6 Common Chiplet Components

6.1 System Control

TMGXQ	NHBRK	CSHRH	WRJNW	NQZFH	FLDNS

6.2 Trusted Security Agent

FFKWC	MWLGW	LDZYP	RCPHY	JMCGT

6.3 System Counter

QVVFX	SZQTS	NQWTT	TQYQP	YGTLQ	KCSBB

Table 2.2: Level 1 Rule Identifiers in the CSA

7 Chiplet Interface Implementation
7.2 Interface Protocols
JCLGK BBMHB PPCHS ZKQNK RGCVB DJWHS WVPQS KKKJS
QYJFV TZRTL CYJBK VHMXL

Table 2.3: Level FULL Rule Identifiers in the CSA

6 Common Chiplet Components
6.1 System Control
BZMMM GGJPQ
6.2 Trusted Security Agent
HRPTP NPXGM VCTWQ PZFPH
7 Chiplet Interface Implementation
7.2 Interface Protocols
KJVCD DPWJP TQRTF VLQZG XBWSM XLNDW NKRPZ NPVYB
TDLVS PXBTX QHGSY LXCQZ DDXMP
7.3 Interface Transports
GJBYZ QDDHW JDCPH CTJXM BGHPZ LQPGR XYZYJ GTDWN
YLBWC HKNHZ VFMQX TWFXJ ZZXCL NCZJN BCQKB YNCSL
CJBHK

Chapter 3
System Types

This chapter describes system type topologies showing the partitioning of a system into chiplets and identifies the chiplet types they contain.

System types are described to ensure a set of chiplets can be used to create a viable system. The system can then be scaled or have functionality added by including more of the mandatory or additional chiplet types.

The intention of the system types in the CSA is to allow any of the chiplet types to be designed independently, and when combined with other chiplets in the system it is ensured to provide the correct set of functions to create a functional system.

The system types are:

- Compute & Hub System.
- Compute Tile System.

3.1 Compute and Hub System

I_{LNZRX} The Compute & Hub System type provides many-core general purpose compute with coherent memory and application specific acceleration, distributed over multiple chiplets:

- The Application PEs are in one or more instances of a Compute 1 Chiplet.
- System main memory, system cache and I/O is provided by a Hub Chiplet.
- Application specific system functions can be added with the following chiplet types:
  - Fully Coherent Expansion Chiplet.
  - I/O Coherent Expansion (Translated) Chiplet.
  - I/O Coherent Expansion (Untranslated) Chiplet.
  - I/O Coherent Expansion (Remote Translation) Chiplet.
  - Fully Coherent Expansion (Remote Translation) Chiplet.
- I/O functions can be partitioned onto separate chiplets using the following types:
  - I/O Chiplet.
  - I/O Controller Chiplet.
- Industry standard PCI or CXL devices can be attached to the system at an appropriately placed interface.

A simplified example of the system is depicted in Figure 3.1.

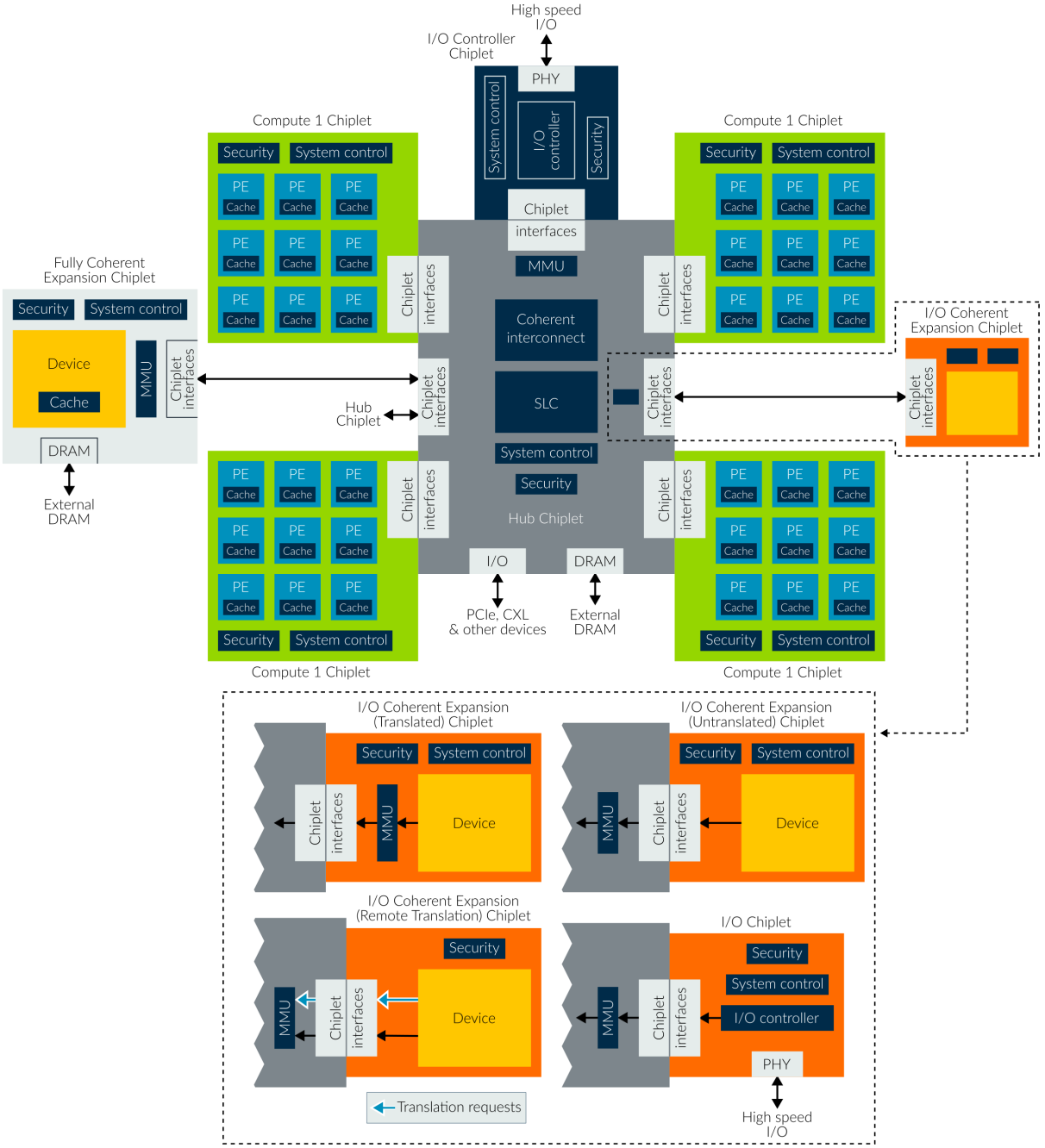

Figure 3.1: Example Compute & Hub System Chiplet Configuration

3.1.1 System Requirements

These requirements apply at the system level. They are collectively and consistently satisfied by all the constituent chiplets.

3.1.1.1 Composition of Chiplet Types

R_{LYHKS} A Compute & Hub System contains one or more Compute 1 Chiplets.

(LEVEL-0)

R_{TFJSM} A Compute & Hub System contains one or more Hub Chiplets.

(LEVEL-0)

R_{LCJNF} A Compute & Hub System contains zero or more Fully Coherent Expansion Chiplets.

(LEVEL-0)

R_{CJZHL} A Compute & Hub System contains zero or more I/O Coherent Expansion (Translated) Chiplets.

(LEVEL-0)

R_{ZCFRY} A Compute & Hub System contains zero or more I/O Coherent Expansion (Untranslated) Chiplets.

(LEVEL-0)

R_{PPLNQ} A Compute & Hub System contains zero or more I/O Coherent Expansion (Remote Translation) Chiplets.

(LEVEL-0)

R_{MDPNF} A Compute & Hub System contains zero or more Fully Coherent Expansion (Remote Translation) Chiplets.

(LEVEL-0)

R_{FZVCL} A Compute & Hub System contains zero or more I/O Chiplets.

(LEVEL-0)

R_{DXYWQ} A Compute & Hub System contains zero or more I/O Controller Chiplets.

(LEVEL-0)

3.1.1.2 Application PE

I_{VQVZW} Typical system use is that Application PEs running under the same operating system are in the same Inner Shareable domain.

R_{GBSPR} All Application PEs in a Compute & Hub System are coherent and in the same Inner Shareable shareability domain.

(LEVEL-0)

I_{WQFNW} In a system implementation containing multiple Application PEs, each Application PE is identified by a unique affinity value reported by the MPIDR_EL1{Aff3, Aff2, Aff1, Aff0} system register.

R_{JLWBK} The assigned value of the MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each Application PE in a Compute & Hub System is unique within the system as a whole.

(LEVEL-0)

3.1.1.3 System Memory

R_{ZKZHH} All system addressable memory is accessible to all Application PEs in a Compute & Hub System.

(LEVEL-0)

I_{ZHVBT} A Compute & Hub System supports a flexible topology of I/O Coherent Expansion Chiplets, I/O Chiplets and I/O Controller Chiplets. All agents in the system access memory through an MMU that provides address translation and memory protection, yet not all types of chiplets contain an MMU. The acceptable location of the MMU is defined by the system physical addressing trust boundary (see Trust Boundaries). Therefore, a system with a valid topology of I/O Coherent Expansion Chiplets, I/O Chiplets and I/O Controller Chiplets must consider the location of each of the MMU components that handle the memory transactions that are initiated by the agents in those chiplets.

R_{DVHYW} System memory transactions that are initiated by agents in a Compute & Hub System outside the physical addressing trust boundary (see Trust Boundaries) are processed through an MMU. This MMU is in the chiplet or chiplets that define(s) the physical addressing trust boundary.

(LEVEL-0)

3.1.1.4 Interrupt Handling

I_{TBTBP} This section concerns interrupts that are targeted at application PEs running the main operating system.

Interrupts in a Compute & Hub System are handled according to the *Arm Generic Interrupt Controller (GIC) Architecture* [4]. The GIC is distributed over multiple chiplets in the system.

R_{QXKKG} All components of the distributed GIC in a Compute & Hub System use the same value for the following fields:

- ICC_CTLR_EL3.A3V
- ICC_CTLR_EL1.A3V
- GICD_TYPER.A3V

(LEVEL-0)

R_{FTXSZ} All components of the distributed GIC in a Compute & Hub System use the same value for the GICR_TYPER.CommonLPIAff field.

(LEVEL-0)

3.1.1.5 System Counter

I_{XFDYD} One or more chiplets in a Compute & Hub System might contain a secondary system counter. A secondary system counter is synchronized to the primary system counter, either directly from the primary counter, or from one secondary system counter to another, from one chiplet to another, forming a sequence back to the primary system counter.

R_{PZWRN} A chiplet in a Compute & Hub System that contains a secondary system counter can be synchronized to the primary system counter through an unbroken sequence of system counters, in different chiplets, back to the primary system counter.

(LEVEL-0)

3.1.1.6 Security

R_{HLQVQ} A Compute & Hub System implements the Secure state (see the Arm ARM [2]). The system supports the access protection of Secure memory.

(LEVEL-0)

I_{SKJJP} A Compute & Hub System might implement the FEAT_RME Arm Architecture [2] feature.

R_{TYGPR} A Compute & Hub System that implements the FEAT_RME Arm Architecture [2] feature is compliant with the Arm Confidential Compute Architecture (CCA) [5]. A system that implements the CCA provides the CCA Security Guarantee to a Realm and the environment that is hosting the Realm.

(LEVEL-0)

3.2 Compute Tile System

I_{RWZCX} The Compute Tile System type provides many-core general purpose compute with coherent memory and application specific acceleration, distributed over multiple chiplets:

- The Application PEs, system main memory, system cache and I/O are in one or more instances of a Compute 2 Chiplet.
- Application specific system functions can be added with the following chiplet types:
  - Fully Coherent Expansion Chiplet.
  - I/O Coherent Expansion (Translated) Chiplet.
  - I/O Coherent Expansion (Untranslated) Chiplet.
  - I/O Coherent Expansion (Remote Translation) Chiplet.
  - Fully Coherent Expansion (Remote Translation) Chiplet.
- I/O functions can be partitioned onto separate chiplets using the following types:
  - I/O Chiplet.
  - I/O Controller Chiplet.
- Industry standard PCI or CXL devices can be attached to the system at an appropriately placed interface.

A simplified example of the system is depicted in Figure 3.2.

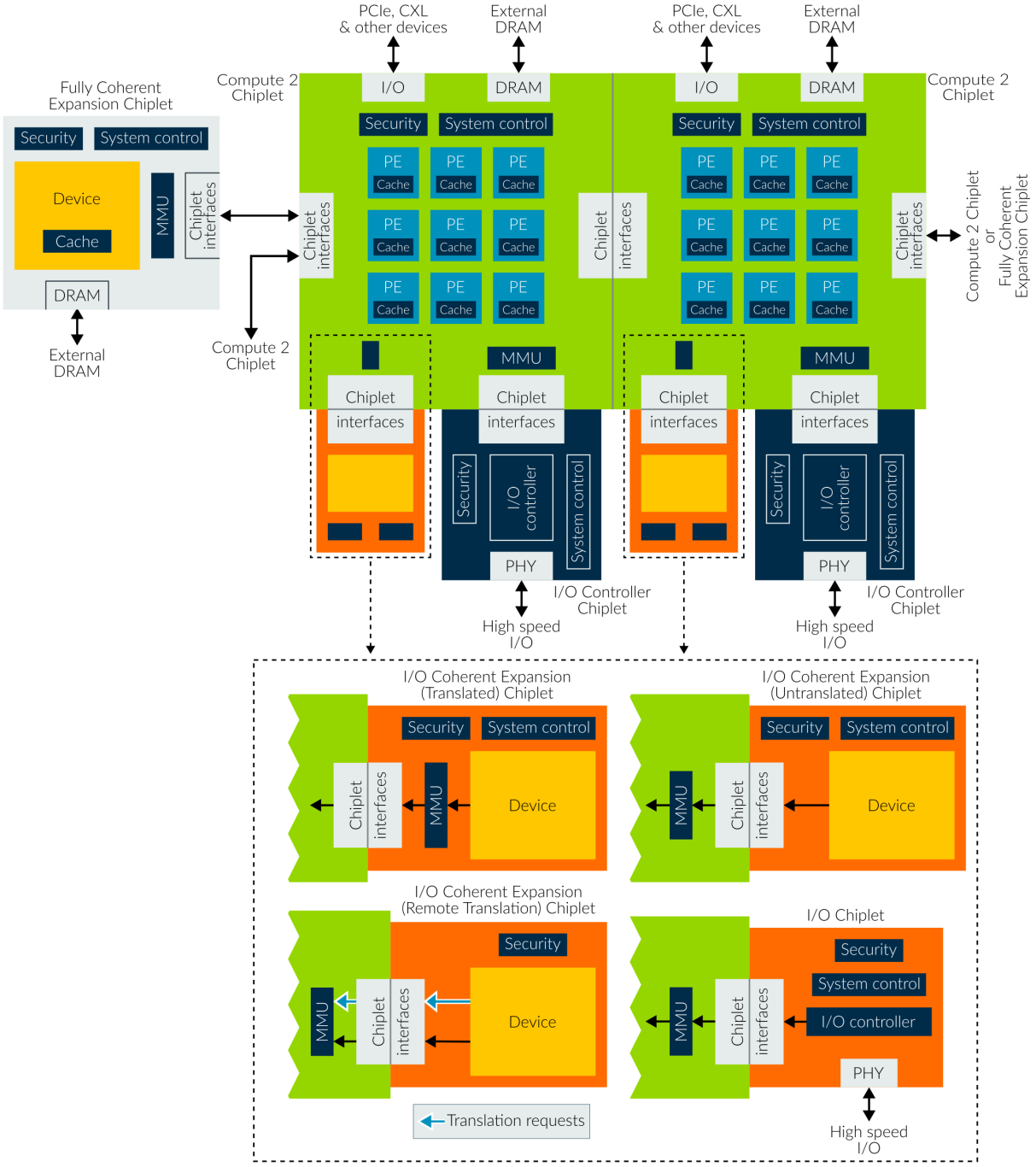

Figure 3.2: Example Compute Tile System Chiplet Configuration

3.2.1 System Requirements

These requirements apply at the system level. They are collectively and consistently satisfied by all of the constituent chiplets.

3.2.1.1 Composition of Chiplet Types

R_{TKPTF} The Compute Tile System contains one or more Compute 2 Chiplets.
(LEVEL-0)

R_{MVWBD} The Compute Tile System contains zero or more Fully Coherent Expansion Chiplets.
(LEVEL-0)

R_{JHWVM} The Compute Tile System contains zero or more I/O Coherent Expansion (Translated) Chiplets.
(LEVEL-0)

R_{DSKBW} The Compute Tile System contains zero or more I/O Coherent Expansion (Untranslated) Chiplets.
(LEVEL-0)

R_{ZFTDV} The Compute Tile System contains zero or more I/O Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{BFSJV} A Compute Tile System contains zero or more Fully Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{ZDVGZ} The Compute Tile System contains zero or more I/O Chiplets.
(LEVEL-0)

R_{CCHPT} The Compute Tile System contains zero or more I/O Controller Chiplets.
(LEVEL-0)

3.2.1.2 Application PE

I_{DLKBG} Typical system use is that Application PEs running under the same operating system are in the same Inner Shareable domain.

R_{WJJVL} All Application PEs in a Compute Tile System are coherent and in the same Inner Shareable shareability domain.
(LEVEL-0)

I_{KJBJK} In a system implementation containing multiple Application PEs, each Application PE is identified by a unique affinity value reported by the MPIDR_EL1{Aff3, Aff2, Aff1, Aff0} system register.

R_{KSQJG} The assigned value of the MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each Application PE in a Compute Tile System is unique within the system as a whole.
(LEVEL-0)

3.2.1.3 System Memory

R_{WTRKM} All system addressable memory is accessible to all Application PEs in a Compute Tile System.
(LEVEL-0)

I_{CNCSN} A Compute Tile System supports a flexible topology of I/O Coherent Expansion Chiplets, I/O Chiplets and I/O Controller Chiplets. All agents in the system access memory through an MMU that provides address translation and memory protection, yet not all types of chiplets contain an MMU. The acceptable location of the MMU is defined by the system physical addressing trust boundary (see Trust Boundaries). Therefore, a system with a valid topology of I/O Coherent Expansion Chiplets, I/O Chiplets and I/O Controller Chiplets must consider the location of each of the MMU components that handle the memory transactions that are initiated by the agents in those chiplets.

R_{VNDCV} System memory transactions that are initiated by agents in a Compute Tile System outside the physical addressing trust boundary (see Trust Boundaries) are processed through an MMU. This MMU is in the chiplet or chiplets that define(s) the physical addressing trust boundary.
(LEVEL-0)

3.2.1.4 Interrupt Handling

I_{FWQWP} This section concerns interrupts that are targeted at application PEs running the main operating system.

Interrupts in a Compute Tile System are handled according to the *Arm Generic Interrupt Controller (GIC) Architecture* [4]. The GIC is distributed over multiple chiplets in the system.

R_{NLRPY} All components of the distributed GIC in a Compute Tile System use the same value for the following fields:

- ICC_CTLR_EL3.A3V,
- ICC_CTLR_EL1.A3V
- GICD_TYPER.A3V

(LEVEL-0)

R_{SCNYP} All components of the distributed GIC in a Compute Tile System use the same value for the GICR_TYPER.CommonLPIAff field.

(LEVEL-0)

3.2.1.5 System Counter

I_{XZCCW} One or more chiplets in a Compute Tile System might contain a secondary system counter. A secondary system counter is synchronized to the primary system counter, either directly from the primary counter, or from one secondary system counter to another, from one chiplet to another, forming a sequence back to the primary system counter.

R_{SHYRP} A chiplet in a Compute Tile System that contains a secondary system counter can be synchronized to the primary system counter through an unbroken sequence of system counters, in different chiplets, back to the primary system counter.

(LEVEL-0)

3.2.1.6 Security

R_{LBCYZ} A Compute Tile System implements the Secure state (see the Arm ARM [2]). The system supports the access protection of Secure memory.

(LEVEL-0)

I_{RTMWG} A Compute Tile System might implement the FEAT_RME Arm Architecture [2] feature.

R_{SMCPF} A Compute Tile System that implements the FEAT_RME Arm Architecture [2] feature is compliant with the Arm Confidential Compute Architecture (CCA) [5]. A system that implements the CCA provides the CCA Security Guarantee to a Realm and the environment that is hosting the Realm.

(LEVEL-0)

Chapter 4
Chiplet Types

This chapter describes chiplet types.

The chiplet types are:

- Compute 1 Chiplet.
- Hub Chiplet.
- Compute 2 Chiplet.
- Fully Coherent Expansion Chiplet.
- Fully Coherent Expansion (Remote Translation) Chiplet.
- I/O Coherent Expansion (Translated) Chiplet.
- I/O Coherent Expansion (Untranslated) Chiplet.
- I/O Coherent Expansion (Remote Translation) Chiplet.
- I/O Chiplet.
- I/O Controller Chiplet.

4.1 Compute 1 Chiplet

IVGHWL

The Compute 1 Chiplet type contains one or more Application PEs, but is not required to contain system main memory or I/O peripherals. The chiplet interfaces with a Hub Chiplet that provides these resources. The chiplet might connect to another similar chiplet.

Simplified examples of a Compute 1 Chiplet are depicted in Figure 4.1.

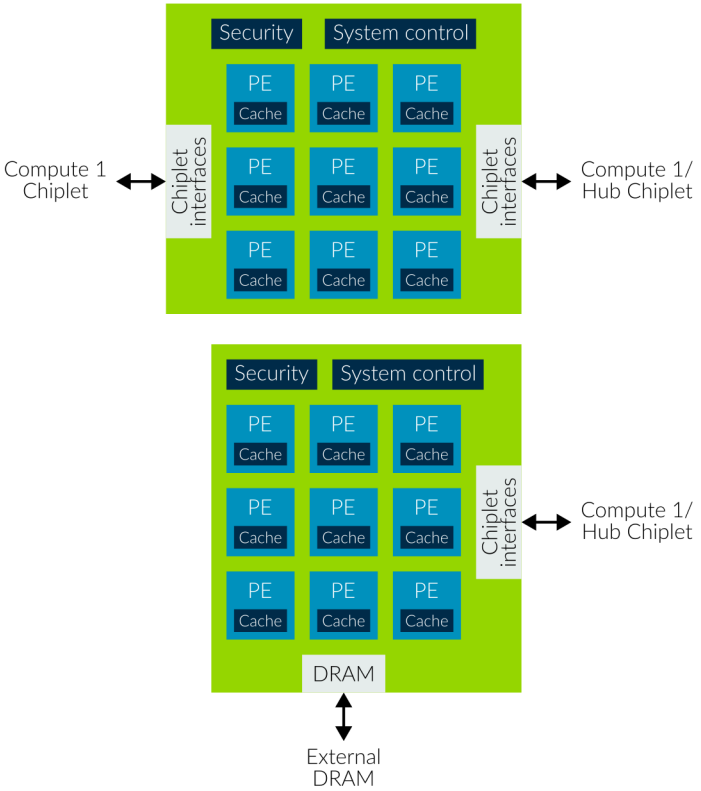

Figure 4.1: Compute 1 Chiplet Examples

The Compute 1 Chiplet is a constituent of the Compute & Hub System system type. Refer to the system type definition for additional system-level specification rules that apply to this chiplet type.

4.1.1 Compute 1 Chiplet Requirements

These requirements are necessary to the definition of the Compute 1 Chiplet type.

4.1.1.1 Chiplet-to-Chiplet Interfaces

R_{BPRJK} A Compute 1 Chiplet is interfaced with zero or more Compute 1 Chiplets.

(LEVEL-0)

R_{RCYDP} A Compute 1 Chiplet is interfaced with zero or more Hub Chiplets.

(LEVEL-0)

R_{JWRCG} A Compute 1 Chiplet is interfaced with at least one of a Hub Chiplets or a Compute 1 Chiplet.

(LEVEL-0)

I_{WRFKN} See Compute 1 Chiplet to Compute 1 Chiplet Interface Requirements.

I_{NBZQX} See Hub Chiplet to Compute Chiplet Interface Requirements.

4.1.1.2 Application PE

R_{XTDHG} A Compute 1 Chiplet contains one or more Application PEs.

(LEVEL-0)

R_{PTZSC} All Application PEs in a Compute 1 Chiplet are fully coherent.

(LEVEL-0)

4.1.1.3 Fully Coherent Device

R_{NCHRQ} A Compute 1 Chiplet does not contain any fully coherent devices.

(LEVEL-0)

4.1.1.4 I/O Coherent Device

R_{KNKSV} A Compute 1 Chiplet does not contain any I/O coherent devices.

(LEVEL-0)

4.1.1.5 I/O Coherent I/O Device

R_{NRJZH} A Compute 1 Chiplet does not contain any I/O coherent I/O devices.

(LEVEL-0)

4.1.1.6 Non-Coherent I/O Device

R_{NGZDK} A Compute 1 Chiplet does not contain any non-coherent I/O devices.

(LEVEL-0)

4.1.1.7 System Memory

R_{KRCKM} A Compute 1 Chiplet might contain Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{XGXZR} The system main memory in or directly connected to a Compute 1 Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{CQDKD} Any coherent caches in a Compute 1 Chiplet receive coherency protocol accesses from any coherent agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{GHRFJ} Any device shared memory in or directly connected to a Compute 1 Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

4.1.1.7.1 MMU Location

I_{PNRLL} Agents in a Compute 1 Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For PEs, this MMU is as specified by the Virtual Memory System Architecture (VMSA) in the Arm ARM [2].

R_{CKTDK} PEs in a Compute 1 Chiplet access system memory through an MMU as specified by the Virtual Memory System Architecture (VMSA) in the Arm ARM [2]. That MMU is contained in the Compute 1 Chiplet.

(LEVEL-0)

I_{YLPWC} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{JBYBY} A Compute 1 Chiplet is within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

4.1.1.8 Interrupt Handling

I_{BFKQR} This section concerns interrupts that are targeted at application PEs running the main operating system.

A Compute 1 Chiplet can be a source and destination of interrupts. Therefore, the chiplet can:

- Generate an interrupt internally, with the destination being a PE in the chiplet.
- Generate an interrupt internally, with the destination being a PE in another chiplet.
- Receive an interrupt from another chiplet, with the destination being a PE in the chiplet.

Interrupts are received at a GIC Distributor (GICD) component in the Compute 1 Chiplet and then:

- If the destination is in the chiplet, forwarded to the targeted PE.
- If the destination is in another chiplet, forwarded to the GICD in the chiplet containing the target PE.

Figure 4.2 shows a simplified representation of the path of an interrupt between chiplets, where both contain a GICD.

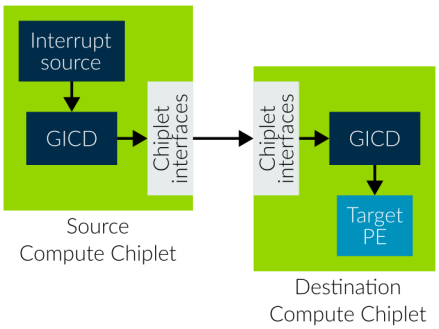

Figure 4.2: Example interrupt communication between two compute chiplets

R_{HDRKM} A Compute 1 Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

U_{VNPCD} The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_{KKYSW} A Compute 1 Chiplet contains a GIC Distributor (GICD).

(LEVEL-0)

R_{MJNJG} The Compute 1 Chiplet GICD receives interrupts from within the same chiplet.

(LEVEL-0)

R_{GVVMS} The Compute 1 Chiplet GICD receives interrupts for the PEs it contains, forwarded from GICDs in other chiplets.

(LEVEL-0)

R_{DZFZV} The Compute 1 Chiplet GICD forwards interrupts to a PE, where the target PE is in the same chiplet.

(LEVEL-0)

R_{ZDSHM} The Compute 1 Chiplet GICD forwards interrupts to the GICD in the chiplet of the target PE, where the target PE is in another chiplet.

(LEVEL-0)

I_{HPCGL} The GIC architecture [4] specifies a *Multiprocessor Affinity Register* MPIDR_EL1 that provides a PE identification mechanism for the purpose of scheduling interrupts. Affinity routing is a hierarchical address-based scheme for identifying specific PEs for interrupt routing. The architecture does not specify at what point the values of MPIDR_EL1 are set. The routing decisions made by the GIC using MPIDR_EL1 are made by software.

R_{FTKXX} The method used to set the value of the *Multiprocessor Affinity Register* MPIDR_EL1 in a GIC in a Compute 1 Chiplet does not produce a conflict with the values of the MPIDR_EL1 registers of other GIC components in other chiplets in the system.

(LEVEL-0)

4.1.1.9 Chiplet System Control

R_{RNSKY} A Compute 1 Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R_{PGGRB} The system controller in a Compute 1 Chiplet is a secondary system controller.

(LEVEL-0)

4.1.1.10 Security

R_{XZXQB} A Compute 1 Chiplet is wholly inside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{NXBYL} Where a Compute 1 Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{MGWMP} Where a Compute 1 Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{XSFHF} A Compute 1 Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{SFPHM} A trusted security agent in a Compute 1 Chiplet is a secondary trusted security agent.

(LEVEL-0)

4.1.1.11 System Counter

I_{QMDNW} A Compute 1 Chiplet contains PEs that require a system counter to support the *Generic Timer* [2].

I_{NPDPJ} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a Compute 1 Chiplet contains a system counter that provides the timestamp value.

R_{ZMQTH} A Compute 1 Chiplet contains a system counter.

(LEVEL-0)

R_{CMQPJ} A system counter in a Compute 1 Chiplet is a secondary system counter.

(LEVEL-0)

4.1.1.12 Debug

I_{KNFPR} A Compute 1 Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{MDGWP} The debug components contained in a Compute 1 Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{QXJHT} A Compute 1 Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.1.1.13 Trace

I_{GNBZV} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{LWGXS} A Compute 1 Chiplet contains one or more CoreSight trace sources and it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the Compute 1 Chiplet or in another chiplet in the system.

(LEVEL-0)

I_{DMMMB} A Compute 1 Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the Compute 1 Chiplet or in another chiplet in the system.

R_{KMBXK} Where a Compute 1 Chiplet is interfaced with a chiplet that contains a source of CoreSight trace data, there is a capability to write the trace data collected from components in chiplets that are directly connected to a Compute 1 Chiplet to the system main memory of a Compute 1 Chiplet or of other chiplets in the system.

(LEVEL-0)

4.1.1.14 System

I_{HRXSM} A unique identifier is necessary for a Compute 1 Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{MRRGT} A Compute 1 Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

4.2 Hub Chiplet

I_CTPLV

The Hub Chiplet type has direct access to system main memory and I/O peripherals. It provides connected compute chiplets with access to system main memory and I/O. It can have multiple connected chiplets providing additional functionality such as non-Application PE fully coherent compute, I/O coherent expansion or I/O.

A Hub Chiplet provides security and system management functions for the system. In a system with multiple instances of a Hub Chiplet, one is designated as the primary and is the coordinator of this functionality.

A simplified example of a Hub Chiplet is depicted in Figure 4.3.

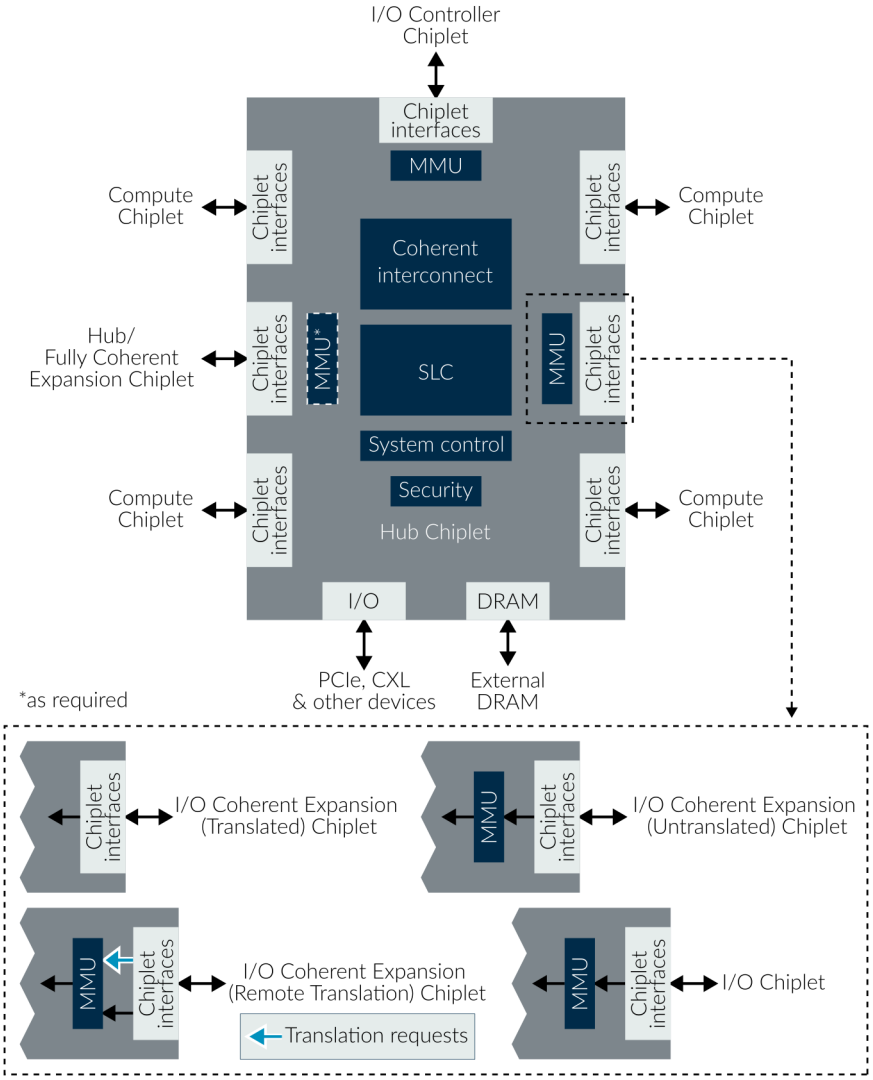

Figure 4.3: Hub Chiplet Example

The Compute 1 Chiplet is a constituent of the Compute & Hub System system type. Refer to the system type definition for additional system-level specification rules that apply to this chiplet type.

4.2.1 Hub Chiplet Requirements

These requirements are necessary to the definition of the Hub Chiplet type.

4.2.1.1 Chiplet-to-Chiplet Interfaces

R_{MZLJX} A Hub Chiplet is interfaced with one or more Compute 1 Chiplets.
(LEVEL-0)

R_{CNQZH} A Hub Chiplet is interfaced with zero or more Compute 2 Chiplets.
(LEVEL-0)

R_{LCQGC} A Hub Chiplet is interfaced with zero or more Hub Chiplets.
(LEVEL-0)

R_{SVWVP} A Hub Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.
(LEVEL-0)

R_{HBYND} A Hub Chiplet is interfaced with zero or more Fully Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{MKYNH} A Hub Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.
(LEVEL-0)

R_{WMTVP} A Hub Chiplet is interfaced with zero or more I/O Coherent Expansion (Untranslated) Chiplets.
(LEVEL-0)

R_{MBQZT} A Hub Chiplet is interfaced with zero or more I/O Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{SPRHJ} A Hub Chiplet is interfaced with zero or more Fully Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{MLQLD} A Hub Chiplet is interfaced with zero or more I/O Chiplets.
(LEVEL-0)

R_{ZTCKN} A Hub Chiplet is interfaced with zero or more I/O Controller Chiplets.
(LEVEL-0)

I_{QPRCW} See Hub Chiplet to Compute Chiplet Interface Requirements.

I_{SDZJJ} See Hub Chiplet to Hub Chiplet Interface Requirements.

I_{NDKNP} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{HSQQH} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

I_{CXHLL} See I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements.

I_{KKNFJ} See I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

I_{FQDZM} See Fully Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

I_{YBZWX} See I/O Chiplet Interface Requirements.

I_{HMTNQ} See I/O Controller Chiplet Interface Requirements.

4.2.1.2 Application PE

R_{PCJMB} A Hub Chiplet does not contain any Application PEs.
(LEVEL-0)

4.2.1.3 Fully Coherent Device

R_{FXFNL} A Hub Chiplet contains zero or more fully coherent devices.
(LEVEL-0)

4.2.1.4 I/O Coherent Device

R_JDYJY A Hub Chiplet contains zero or more I/O coherent devices.

(LEVEL-0)

4.2.1.5 I/O Coherent I/O Device

R_PMKFN A Hub Chiplet contains zero or more I/O coherent I/O devices.

(LEVEL-0)

4.2.1.6 Non-Coherent I/O Device

R_YHQDR A Hub Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.2.1.7 System Memory

R_YVFHS A Hub Chiplet contains Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_RXLZG The system main memory in or directly connected to a Hub Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_MTBPW Any system addressable memory in the system is available through a Hub Chiplet to agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_VJFWY Any coherent caches in a Hub Chiplet accept coherency protocol accesses from any coherent agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_KZFYG Any device shared memory in or directly connected to a Hub Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

4.2.1.7.1 MMU Location

I_HXRBV Agents in a Hub Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].
- For Fully-Coherent, non-PE agents, this MMU is IMPLEMENTATION DEFINED.

R_FQRZT I/O coherent or non-coherent agents in a Hub Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is contained in the Hub Chiplet.

(LEVEL-0)

R_JNXMC Fully coherent agents in a Hub Chiplet access system memory through an IMPLEMENTATION DEFINED MMU. That MMU is contained in the Hub Chiplet.

(LEVEL-0)

I_NFNQX The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_MSJZQ A Hub Chiplet is within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{MPFHT} Where a Hub Chiplet is directly connected to a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries), that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_{NRZBH} Memory accesses that are initiated by agents in a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries) and are received by a Hub Chiplet are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the Hub Chiplet coincides with crossing to within the system physical addressing trust boundary, this MMU is in the Hub Chiplet.

(LEVEL-0)

4.2.1.8 Interrupt Handling

I_{MWPTG} This section concerns interrupts that are targeted at application PEs running the main operating system.

Interrupts in a Hub Chiplet are handled according to the *Arm Generic Interrupt Controller (GIC) Architecture* [4].

A Hub Chiplet can be a source of interrupts, or convey interrupts from other chiplets. Therefore, the chiplet can:

- Generate an interrupt internally, with the destination being a PE in another chiplet.
- Receive an interrupt from another chiplet, with the destination being a PE in another chiplet.

The components in a Hub Chiplet can generate interrupts, or interrupts can be received from other chiplets that do not have a GIC distributor (GICD) component. These interrupts are received at the Hub Chiplet GICD and then forwarded to the GICD in the chiplet that contains the target PE.

Figure 4.4 shows a simplified representation of the path of an interrupt between chiplets, where both contain a GICD.

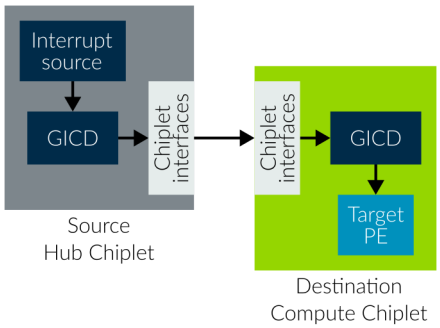

Figure 4.4: Example interrupt communication between hub and compute chiplets

U_{GQTFP} The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_{PXHHJ} A Hub Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

R_{FZDYK} A Hub Chiplet contains a GIC Distributor (GICD).

(LEVEL-0)

R_{FTXZM} The Hub Chiplet GICD receives interrupts from within the same chiplet.

(LEVEL-0)

R_{ZPLGW} The Hub Chiplet GICD receives interrupts from other chiplets in the system, that do not contain a GICD.

(LEVEL-0)

R_{VNGDP} The Hub Chiplet GICD forwards interrupts to the GICD in the chiplet of the target PE, where the target PE is in another chiplet.

(LEVEL-0)

R_{SDDYH} If the Hub Chiplet receives MSIs from other connected chiplets, it contains an Interrupt Translation Service (ITS) component.

(LEVEL-0)

I_{RKGCG} The GIC architecture [4] specifies a *Multiprocessor Affinity Register* MPIDR_EL1 that provides a PE identification mechanism for the purpose of scheduling interrupts. Affinity routing is a hierarchical address-based scheme for identifying specific PEs for interrupt routing. The architecture does not specify at what point the values of MPIDR_EL1 are set. The routing decisions made by the GIC using MPIDR_EL1 are made by software.

R_{XLLCC} The method used to set the value of the *Multiprocessor Affinity Register* MPIDR_EL1 in a GIC in a Hub Chiplet does not produce a conflict with the values of the MPIDR_EL1 registers of other GIC components in other chiplets in the system.

(LEVEL-0)

4.2.1.9 Non-volatile Memory

R_{RFPQN} A Hub Chiplet contains non-volatile memory, either integrated or directly connected, that can hold boot code images for the trusted security agents, system controllers and Application PEs in the system.

(LEVEL-0)

R_{NJQDV} When a Hub Chiplet is designated as the primary chiplet (see System), it contains non-volatile memory, either integrated or directly connected, that can hold boot code for the trusted security agents, system controllers and Application PEs in the system.

(LEVEL-0)

R_{NXDVW} When a Hub Chiplet is not designated as the primary chiplet (see System), it might contain non-volatile memory, either integrated or directly connected, that can hold boot code for the trusted security agents, system controllers and Application PEs in the system.

(LEVEL-0)

4.2.1.10 Chiplet System Control

R_{CKXYX} A Hub Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R_{ZFWWC} When a Hub Chiplet is designated as the primary chiplet (see System), the system controller it contains is the primary system controller.

(LEVEL-0)

R_{PYRRZ} When a Hub Chiplet is not designated as the primary chiplet (see System), the system controller it contains is a secondary system controller.

(LEVEL-0)

4.2.1.11 Security

R_{QCBRF} A Hub Chiplet is wholly inside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{PLXZG} Where a Hub Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{XSBBL} Where a Hub Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{ZCPHB} A Hub Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.
- Secure storage of immutable boot code.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{HCCDW} Where a Hub Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Creation of a CCA platform attestation token.
- CCA enforced key derivation.
- Capture of measurements, with hash extend and lock semantics.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{LDSVP} When a Hub Chiplet is designated as the primary chiplet (see System), a single trusted security agent it contains is the primary trusted security agent and any others are secondary trusted security agents.

(LEVEL-0)

R_{GYSJB} When a Hub Chiplet is not designated as the primary chiplet (see System), all of the trusted security agents it contains are secondary trusted security agents.

(LEVEL-0)

4.2.1.12 System Counter

I_{XYJRP} When a Hub Chiplet is designated as the primary chiplet (see System), it contains the primary system counter to which the secondary system counters in the system are synchronized.

I_{QHJQM} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a Hub Chiplet contains a system counter that provides the timestamp value.

R_{BJBSF} A Hub Chiplet contains a system counter.

(LEVEL-0)

R_{QPDWG} When a Hub Chiplet is designated as the primary chiplet (see System), the system counter it contains is the primary system counter.

(LEVEL-0)

R_{LGWZM} When a Hub Chiplet is not designated as the primary chiplet (see System), the system counter it contains is a secondary system counter.

(LEVEL-0)

4.2.1.13 Debug

I_{DJTMK} A Hub Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{SXNDJ} The debug components contained in a Hub Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{TTRSK} A Hub Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.2.1.14 Trace

I_{XQFQN} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{GNBZX} A Hub Chiplet contains one or more CoreSight trace sources and it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the Hub Chiplet or in another chiplet in the system.

(LEVEL-0)

I_{DPTFV} A Hub Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the Hub Chiplet or in another chiplet in the system.

R_{GHVLC} There is a capability to write the trace data collected from components in chiplets that are directly connected to a Hub Chiplet to the system main memory of a Hub Chiplet or of other chiplets in the system.

(LEVEL-0)

4.2.1.15 System

I_{WMGRV} A unique identifier is necessary for a Hub Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{YJBND} A Hub Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

R_{KYYNQ} There is a single chiplet in the system that is designated as the *primary* chiplet and it is an instance of a Hub Chiplet.

(LEVEL-0)

R_{DZGQK} When there are two or more instances of a Hub Chiplet in the system, only one of them is designated as the *primary* chiplet. A Hub Chiplet contains an IMPLEMENTATION DEFINED mechanism that determines if it is the *primary* or a *secondary* chiplet.

(LEVEL-0)

U_{ZCTBJ} An example mechanism for determining the primary chiplet is an input signal that is tied-off at the package.

4.3 Compute 2 Chiplet

IXBBNM

The Compute 2 Chiplet type contains one or more Application PEs and has direct access to system main memory and I/O peripherals. It can have multiple connected chiplets providing additional functionality, such as another Compute 2 Chiplet, non-Application PE fully coherent compute, I/O coherent expansion or I/O.

A Compute 2 Chiplet provides security and system management functions for the system. In a system with multiple instances of a Compute 2 Chiplet, one is designated as the primary and is the coordinator of this functionality.

A simplified example of a Compute 2 Chiplet is depicted in Figure 4.5.

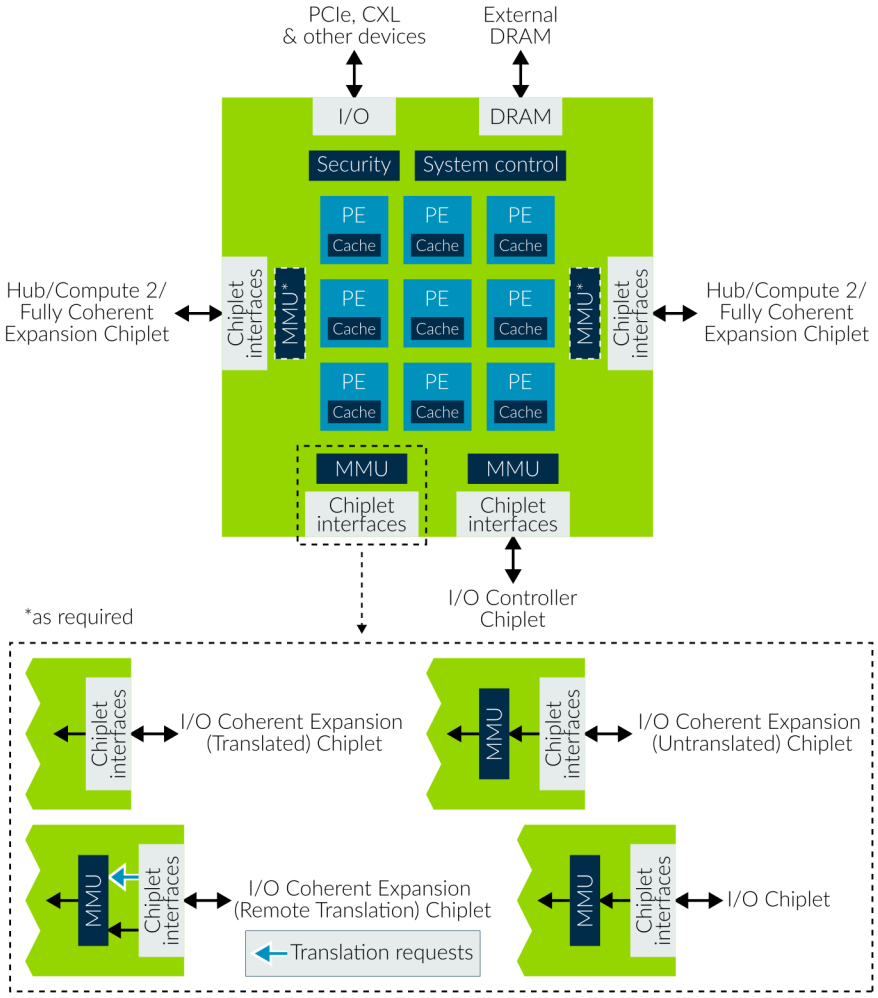

Figure 4.5: Compute 2 Chiplet Example

The Compute 2 Chiplet is a constituent of the Compute Tile System system type. Refer to the system type definition for additional system-level specification rules that apply to the chiplet type.

4.3.1 Compute 2 Chiplet Requirements

These requirements are necessary to the definition of the Compute 2 Chiplet type.

4.3.1.1 Chiplet-to-Chiplet Interfaces

R_{SJNGN} A Compute 2 Chiplet is interfaced with zero or more Hub Chiplets.
(LEVEL-0)

R_{NDHDT} A Compute 2 Chiplet is interfaced with zero or more Compute 2 Chiplets.
(LEVEL-0)

R_{KNHMM} A Compute 2 Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.
(LEVEL-0)

R_{CGKQW} A Compute 2 Chiplet is interfaced with zero or more Fully Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{MDFWD} A Compute 2 Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.
(LEVEL-0)

R_{WLHVZ} A Compute 2 Chiplet is interfaced with zero or more I/O Coherent Expansion (Untranslated) Chiplets.
(LEVEL-0)

R_{JZSRC} A Compute 2 Chiplet is interfaced with zero or more I/O Coherent Expansion (Remote Translation) Chiplets.
(LEVEL-0)

R_{DVNHD} A Compute 2 Chiplet is interfaced with zero or more I/O Chiplets.
(LEVEL-0)

R_{QGGWQ} A Compute 2 Chiplet is interfaced with zero or more I/O Controller Chiplets.
(LEVEL-0)

I_{ZQGLZ} See Hub Chiplet to Compute Chiplet Interface Requirements.

I_{VXZQN} See Compute 2 Chiplet to Compute 2 Chiplet Interface Requirements.

I_{LVBGP} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{SPPPT} See Fully Coherent Expansion Chiplet (Remote Translation) Interface Requirements.

I_{XQGLB} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

I_{HJRVL} See I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements.

I_{DBDNY} See I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

I_{MHXYJ} See I/O Chiplet Interface Requirements.

I_{LNRCG} See I/O Controller Chiplet Interface Requirements.

4.3.1.2 Application PE

R_{LGXLN} A Compute 2 Chiplet contains one or more Application PEs.
(LEVEL-0)

R_{XNYDJ} All Application PEs in a Compute 2 Chiplet are fully coherent.
(LEVEL-0)

4.3.1.3 Fully Coherent Device

R_{PHHQM} A Compute 2 Chiplet contains zero or more fully coherent devices.
(LEVEL-0)

4.3.1.4 I/O Coherent Device

R_{ZVZZN} A Compute 2 Chiplet contains zero or more I/O coherent devices.
(LEVEL-0)

4.3.1.5 I/O Coherent I/O Device

R_{VYJFL} A Compute 2 Chiplet contains zero or more I/O coherent I/O devices.

(LEVEL-0)

4.3.1.6 Non-Coherent I/O Device

R_{ZCLCN} A Compute 2 Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.3.1.7 System Memory

R_{FXTRR} A Compute 2 Chiplet contains Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{XCVGN} The system main memory in or directly connected to a Compute 2 Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{ZXWKH} Any system addressable memory in the system is available through a Compute 2 Chiplet to agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{DZDHQ} Any coherent caches in a Compute 2 Chiplet accept coherency protocol accesses from any coherent agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{PBYQY} Any device shared memory in or directly connected to a Compute 2 Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

4.3.1.7.1 MMU Location

I_{VXGYS} Agents in a Compute 2 Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For PEs, this MMU is as specified by the Virtual Memory System Architecture (VMSA) in the Arm ARM [2].
- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_{BBBJT} PEs in a Compute 2 Chiplet access system memory through an MMU as specified by the Virtual Memory System Architecture (VMSA) in the Arm ARM [2]. That MMU is contained in the chiplet.

(LEVEL-0)

R_{RWFFF} I/O coherent or non-coherent agents in a Compute 2 Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is contained in the chiplet.

(LEVEL-0)

I_{CGSLB} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{KYSDY} A Compute 2 Chiplet is within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{VGCBH} Where a Compute 2 Chiplet is directly connected to a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries), that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_MRYYB Memory accesses that are initiated by agents in a chiplet that are not within the system physical addressing trust boundary (see Trust Boundaries) and are received by a Compute 2 Chiplet are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the Compute 2 Chiplet coincides with crossing to within the system physical addressing trust boundary, that MMU is in the Compute 2 Chiplet.

(LEVEL-0)

4.3.1.8 Interrupt Handling

I_KNLTT This section concerns interrupts that are targeted at application PEs running the main operating system.

A Compute 2 Chiplet can be a source and destination of interrupts. Therefore, the chiplet can:

- Generate an interrupt internally, with the destination being a PE in the chiplet.
- Generate an interrupt internally, with the destination being a PE in another chiplet.
- Receive an interrupt from another chiplet, with the destination being a PE in the chiplet.
- Receive an interrupt from another chiplet, with the destination being a PE in another chiplet.

The components in a Compute 2 Chiplet can generate interrupts, or interrupts are received from other chiplets that do not have a GIC distributor (GICD) component. Interrupts are received at a GIC distributor (GICD) component in the Compute 2 Chiplet and then:

- If the destination is in the chiplet, forwarded to the target PE.
- If the destination is in another chiplet, forwarded to the GICD in the chiplet that contains the target PE.

Figure 4.6 shows a simplified representation of the path of an interrupt between chiplets, where both contain a GICD.

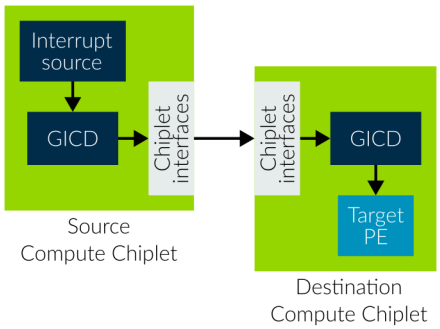

Figure 4.6: Example interrupt communication between two compute chiplets

R_XBBLT A Compute 2 Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

U_JCSNK The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_LYFPT A Compute 2 Chiplet contains a GIC Distributor (GICD).

(LEVEL-0)

R_THPPG The Compute 2 Chiplet GICD receives interrupts from within the same chiplet.

(LEVEL-0)

R_NYZVM The Compute 2 Chiplet GICD receives interrupts from other chiplets in the system, that do not contain a GICD.

(LEVEL-0)

R_YNMLX The Compute 2 Chiplet GICD receives interrupts for the PEs it contains, forwarded from other GICDs in other chiplets.

(LEVEL-0)

R_SGZXK The Compute 2 Chiplet GICD forwards interrupts to a PE, where the target PE is in the same chiplet.

(LEVEL-0)

R_VVDFD The Compute 2 Chiplet GICD forwards interrupts to the GICD in the chiplet of the target PE, where the target PE is in another chiplet.

(LEVEL-0)

R_SCYSD If the Compute 2 Chiplet receives MSIs from other chiplets with which it is connected, it contains an Interrupt Translation Service (ITS) component.

(LEVEL-0)

I_VGJJL The GIC architecture [4] specifies a *Multiprocessor Affinity Register* MPIDR_EL1 that provides a PE identification mechanism for the purpose of scheduling interrupts. Affinity routing is a hierarchical address-based scheme for identifying specific PEs for interrupt routing. The architecture does not specify at what point the values of MPIDR_EL1 are set. The routing decisions made by the GIC using MPIDR_EL1 are made by software.

R_VVNTG The method used to set the value of the *Multiprocessor Affinity Register* MPIDR_EL1 in a GIC in a Compute 2 Chiplet does not produce a conflict with the values of the MPIDR_EL1 registers of other GIC components in other chiplets in the system.

(LEVEL-0)

4.3.1.9 Non-volatile Memory

R_NLGDR When a Compute 2 Chiplet is designated as the primary chiplet (see System), it contains non-volatile memory, either integrated or directly connected, that can hold boot code images for the trusted security agents, system controllers and Application PEs in the system.

(LEVEL-0)

R_WVMSK When a Compute 2 Chiplet is not designated as the primary chiplet (see System), it might contain non-volatile memory, either integrated or directly connected, that can hold boot code images for the trusted security agents, system controllers and Application PEs in the system.

(LEVEL-0)

4.3.1.10 Chiplet System Control

R_PGTMQ A Compute 2 Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R_YQMHG When a Compute 2 Chiplet is designated as the primary chiplet (see System), the system controller it contains is the primary system controller.

(LEVEL-0)

R_FDRRV When a Compute 2 Chiplet is not designated as the primary chiplet (see System), the system controller it contains is a secondary system controller.

(LEVEL-0)

4.3.1.11 Security

R_MSCJJ A Compute 2 Chiplet is wholly inside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_RXMNK Where a Compute 2 Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_LGHFC Where a Compute 2 Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{FWCJG} A Compute 2 Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.
- Secure storage of immutable boot code.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{FZDJN} Where a Compute 2 Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Creation of a CCA platform attestation token.
- CCA enforced key derivation.
- Capture of measurements, with hash extend and lock semantics.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{KGFNL} When a Compute 2 Chiplet is designated as the primary chiplet (see System), a single trusted security agent it contains is the primary trusted security agent and any others are secondary trusted security agents.

(LEVEL-0)

R_{GNZNV} When a Compute 2 Chiplet is not designated as the primary chiplet (see System), all of the trusted security agents it contains are secondary trusted security agents.

(LEVEL-0)

4.3.1.12 System Counter

I_{YCYQL} A Compute 2 Chiplet contains PEs that require a system counter to support the *Generic Timer* [2].

I_{ZJTMK} When a Compute 2 Chiplet is designated as the primary chiplet (see System), it contains the primary system counter to which the secondary system counters in the system are synchronized.

I_{CJKBP} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a Compute 2 Chiplet contains a system counter that provides the timestamp value.

R_{BGJCM} A Compute 2 Chiplet contains a system counter.

(LEVEL-0)

R_{QHPNB} When a Compute 2 Chiplet is designated as the primary chiplet (see System), the system counter it contains is the primary system counter.

(LEVEL-0)

R_{VRVYH} When a Compute 2 Chiplet is not designated as the primary chiplet (see System), the system counter it contains is a secondary system counter.

(LEVEL-0)

4.3.1.13 Debug

I_{ZNNHD} A Compute 2 Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{ZDBLR} The debug components contained in a Compute 2 Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{CJQRD} A Compute 2 Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.3.1.14 Trace

I_{ZKVPW} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{CKWDT} A Compute 2 Chiplet contains one or more CoreSight trace sources and it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the Compute 2 Chiplet or in another chiplet in the system.

(LEVEL-0)

I_{YXHXJ} A Compute 2 Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the Compute 2 Chiplet or in another chiplet in the system.

R_{DPXXH} Where a Compute 2 Chiplet is interfaced with a chiplet that contains a source of CoreSight trace data, there is a capability to write the trace data collected from components in chiplets that are directly connected to a Compute 2 Chiplet to the system main memory of a Compute 2 Chiplet or of other chiplets in the system.

(LEVEL-0)

4.3.1.15 System

I_{DBTLF} A unique identifier is necessary for a Compute 2 Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{XDDFR} A Compute 2 Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

R_{XYVWT} There is a single chiplet in the system that is designated as the *primary* chiplet. When there are one or more instances of a Hub Chiplet in the system a single instance of a Hub Chiplet is designated as the *primary* chiplet. When there are zero instances of a Hub Chiplet in the system a single instance of a Compute 2 Chiplet is designated as the *primary* chiplet.

(LEVEL-0)

R_{LPTTQ} When there are zero instances of a Hub Chiplet and two or more instances of a Compute 2 Chiplet in the system, only one of the instances of a Compute 2 Chiplet is designated as *primary*. A Compute 2 Chiplet contains an IMPLEMENTATION DEFINED mechanism that determines if it is a *primary* or a *secondary* chiplet.

(LEVEL-0)

U_{TYRVJ} An example mechanism for determining the primary chiplet is an input signal that is tied-off at the package.

4.4 Fully Coherent Expansion Chiplet

I_{TLNDW} The Fully Coherent Expansion Chiplet type connects with a host system to provide fully coherent application specific acceleration and access to system main memory. The chiplet might include interfaces to external DRAM. The chiplet might include interfaces with other fully coherent, I/O coherent or non-coherent chiplet types.

Simplified examples of a Fully Coherent Expansion Chiplet are depicted in Figure 4.7.

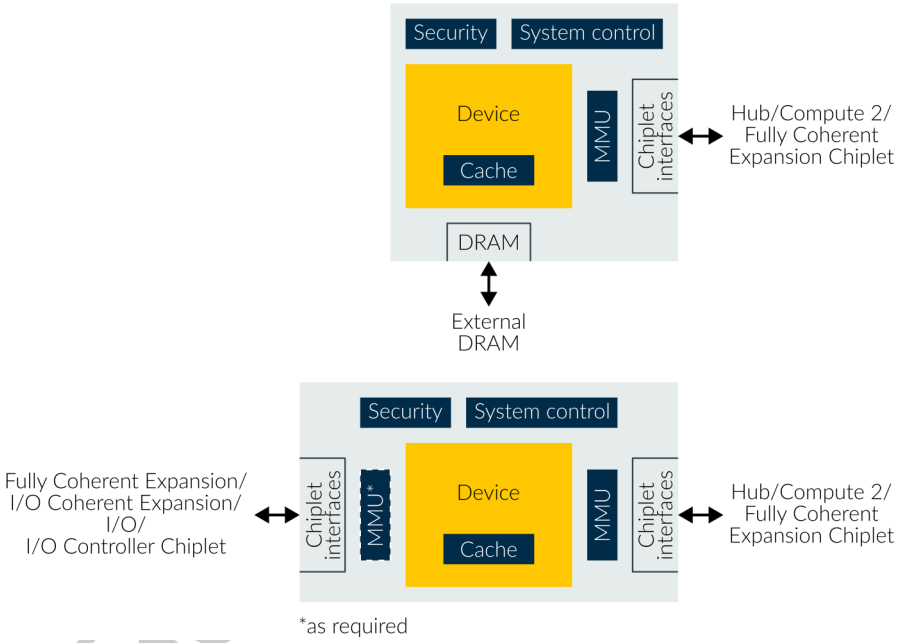

Figure 4.7: Fully Coherent Expansion Chiplet Examples

A Fully Coherent Expansion Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.4.1 Fully Coherent Expansion Chiplet Requirements

These requirements are necessary to the definition of the Fully Coherent Expansion Chiplet type.

4.4.1.1 Chiplet-to-Chiplet Interfaces

R_{SHXTT} A Fully Coherent Expansion Chiplet is interfaced with zero or more of either:

- A Hub Chiplet.
- A Compute 2 Chiplet.

(LEVEL-0)

R_{MKCTS} A Fully Coherent Expansion Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.
(LEVEL-0)

R_{HMZJQ} A Fully Coherent Expansion Chiplet is interfaced with zero or more Fully Coherent Expansion (Remote Translation) Chiplets.

(LEVEL-0)

R_{YDWVV} A Fully Coherent Expansion Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.
(LEVEL-0)

R_{NBXWR} A Fully Coherent Expansion Chiplet is interfaced with zero or more I/O Coherent Expansion (Untranslated) Chiplets.

(LEVEL-0)

R_{FMBVG} A Fully Coherent Expansion Chiplet is interfaced with zero or more I/O Coherent Expansion (Remote Translation) Chiplets.

(LEVEL-0)

R_{JNCKH} A Fully Coherent Expansion Chiplet is interfaced with zero or more I/O Chiplets.

(LEVEL-0)

R_{VCPQR} A Fully Coherent Expansion Chiplet is interfaced with zero or more I/O Controller Chiplets.

(LEVEL-0)

I_{YVRSB} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{GPVMH} See Fully Coherent Expansion Chiplet (Remote Translation) Interface Requirements.

I_{HVWGS} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

I_{TRFHV} See I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements.

I_{SVFPP} See I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

I_{HVJYJ} See I/O Chiplet Interface Requirements.

I_{MQPDR} See I/O Controller Chiplet Interface Requirements.

4.4.1.2 Application PE

R_{FJDNP} A Fully Coherent Expansion Chiplet does not contain any Application PEs.

(LEVEL-0)

4.4.1.3 Fully Coherent Device

R_{SPTFW} A Fully Coherent Expansion Chiplet contains one or more fully coherent devices.

(LEVEL-0)

4.4.1.4 I/O Coherent Device

R_{FQPML} A Fully Coherent Expansion Chiplet contains zero or more I/O coherent devices.

(LEVEL-0)

4.4.1.5 I/O Coherent I/O Device

R_{QDMPX} A Fully Coherent Expansion Chiplet contains zero or more I/O coherent I/O devices.

(LEVEL-0)

4.4.1.6 Non-Coherent I/O Device

R_{LYVBX} A Fully Coherent Expansion Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.4.1.7 System Memory

R_{DWFSK} A Fully Coherent Expansion Chiplet might contain Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{XVWQF} Any system main memory in a Fully Coherent Expansion Chiplet receives accesses by agents in the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{QQTXS} Any coherent caches in a Fully Coherent Expansion Chiplet receive coherency protocol accesses from any coherent agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{RDGJV} Any device shared memory in or directly connected to a Fully Coherent Expansion Chiplet receives accesses by agents within the chiplet or in other chiplets in the system.

(LEVEL-0)

4.4.1.7.1 MMU Location

I_{RSWCG} Agents in a Fully Coherent Expansion Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].
- For Fully-Coherent, non-PE agents, this MMU is IMPLEMENTATION DEFINED.

R_{LDYRT} I/O coherent or non-coherent agents in a Fully Coherent Expansion Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is contained in the chiplet.

(LEVEL-0)

R_{CXLDV} Fully coherent agents in a Fully Coherent Expansion Chiplet access system memory through an IMPLEMENTATION DEFINED MMU. That MMU is contained in the Fully Coherent Expansion Chiplet.

(LEVEL-0)

I_{MCWLG} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{DDBRW} A Fully Coherent Expansion Chiplet is within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{NPQTP} Where a Fully Coherent Expansion Chiplet is directly connected to a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries), that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_{FPNKN} Memory accesses that are initiated by agents in a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries) and are received by a Fully Coherent Expansion Chiplet are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the Fully

Coherent Expansion Chiplet coincides with crossing to within the system physical addressing trust boundary, this MMU is in the Fully Coherent Expansion Chiplet.

(LEVEL-0)

4.4.1.8 Interrupt Handling

I_FJGYD This section concerns interrupts that are targeted at application PEs running the main operating system.

Interrupts in a Fully Coherent Expansion Chiplet are handled according to the *Arm Generic Interrupt Controller (GIC) Architecture* [4].

A Fully Coherent Expansion Chiplet can be a source of interrupts, or convey interrupts from other chiplets. Therefore, the chiplet can:

- Generate an interrupt internally, with the destination being a PE in another chiplet.
- Receive an interrupt from another chiplet, with the destination being a PE in another chiplet.

The components in a Fully Coherent Expansion Chiplet can generate interrupts, or interrupts can be received from other chiplets that do not have a GIC distributor (GICD) component. These interrupts are either :

- If the chiplet has a GICD, received at the Fully Coherent Expansion Chiplet GICD and then forwarded to the GICD in the chiplet that contains the target PE.
- If the chiplet does not have a GICD, forwarded to a connected chiplets GICD or ITS, dependent on interrupt type.

Figure 4.8 shows a simplified representation of the path of an interrupt between chiplets, where both contain a GICD.

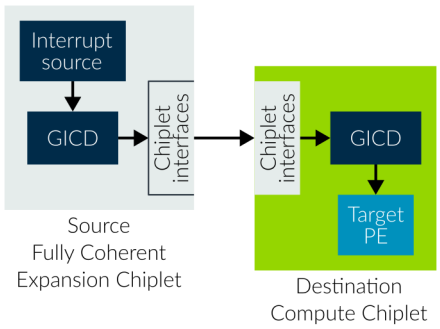

Figure 4.8: Example interrupt communication between fully coherent expansion and compute chiplets

R_CSCZG A Fully Coherent Expansion Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

U_LGMCX The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_MCDLS A Fully Coherent Expansion Chiplet can contain a GIC Distributor (GICD).

(LEVEL-0)

R_ZRPHF When a Fully Coherent Expansion Chiplet contains a GICD, the GICD receives interrupts from within the same chiplet.

(LEVEL-0)

R_QZSYG When a Fully Coherent Expansion Chiplet contains a GICD, the GICD can receive interrupts from other chiplets in the system, that do not contain a GICD.

(LEVEL-0)

R_{PWHNJ} When a Fully Coherent Expansion Chiplet contains a GICD, where the target PE of an interrupt is in another chiplet, the GICD forwards the interrupt to the GICD in the chiplet of the target PE.

(LEVEL-0)

R_{MBCXY} When a Fully Coherent Expansion Chiplet contains a GICD, if the Fully Coherent Expansion Chiplet receives MSIs from connected chiplets, it contains an Interrupt Translation Service (ITS) component.

(LEVEL-0)

I_{SMDQD} The GIC architecture [4] specifies a *Multiprocessor Affinity Register* MPIDR_EL1 that provides a PE identification mechanism for the purpose of scheduling interrupts. Affinity routing is a hierarchical address-based scheme for identifying specific PEs for interrupt routing. The architecture does not specify at what point the values of MPIDR_EL1 are set. The routing decisions made by the GIC using MPIDR_EL1 are made by software.

R_{QCHVW} When a Fully Coherent Expansion Chiplet contains a GICD, the method used to set the value of the *Multiprocessor Affinity Register* MPIDR_EL1 in GIC components in a Fully Coherent Expansion Chiplet does not conflict with the values of the MPIDR_EL1 registers of other GIC components in other chiplets in the system.

(LEVEL-0)

R_{FVWGK} When a Fully Coherent Expansion Chiplet does not contain a GICD, it transmits LPIs as MSIs to a GIC ITS in a connected chiplet.

(LEVEL-0)

R_{VRPMH} When a Fully Coherent Expansion Chiplet does not contain a GICD, it transmits SPI interrupts as IMPLEMENTATION DEFINED messages to a GICD in a connected chiplet.

(LEVEL-0)

4.4.1.9 Chiplet System Control

R_{ZNXXR} A Fully Coherent Expansion Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R_{RJMNL} The system controller in a Fully Coherent Expansion Chiplet is a secondary system controller.

(LEVEL-0)

4.4.1.10 Security

R_{CVMFD} A Fully Coherent Expansion Chiplet is wholly inside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{PDTQR} Where a Fully Coherent Expansion Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{BXZMJ} Where a Fully Coherent Expansion Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the MMU it contains supports the Granule Protection Check (GPC) and makes Root accesses to the Granule Protection Table (GPT).

R_{ZKPSD} Where a Fully Coherent Expansion Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{FFWWY} A Fully Coherent Expansion Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_BMPMM A trusted security agent in a Fully Coherent Expansion Chiplet is a secondary trusted security agent.
(LEVEL-0)

4.4.1.11 System Counter

I_NSLCW CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a Fully Coherent Expansion Chiplet might contain a system counter that provides the timestamp value.

R_FCCGZ Where a Fully Coherent Expansion Chiplet contains a source of CoreSight trace data, it contains a system counter.
(LEVEL-0)

R_HCQGD A system counter in a Fully Coherent Expansion Chiplet is a secondary system counter.
(LEVEL-0)

4.4.1.12 Debug

I_TCZKV A Fully Coherent Expansion Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_ZDSMN The debug components contained in a Fully Coherent Expansion Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.
(LEVEL-0)

R_CMTXV A Fully Coherent Expansion Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.
(LEVEL-0)

4.4.1.13 Trace

I_BVRNM The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_ZJLZD Where a Fully Coherent Expansion Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the Fully Coherent Expansion Chiplet or in another chiplet in the system.
(LEVEL-0)

I_SMWJV A Fully Coherent Expansion Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the Fully Coherent Expansion Chiplet or in another chiplet in the system.

R_BDDYL Where a Fully Coherent Expansion Chiplet is interfaced with a chiplet that contains a source of CoreSight trace data, there is a capability to write the trace data collected from components in chiplets that are directly connected to a Fully Coherent Expansion Chiplet to the system main memory of a Fully Coherent Expansion Chiplet or of other chiplets in the system.
(LEVEL-0)

4.4.1.14 System

I_TRCQC A unique identifier is necessary for a Fully Coherent Expansion Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_VJGGP A Fully Coherent Expansion Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.
(LEVEL-0)

4.5 Fully Coherent Expansion (Remote Translation) Chiplet

Note

Arm is requesting feedback on the need for this chiplet type. Arm currently recommends using the Fully Coherent Expansion Chiplet for flexibility and IP support.

IXBDVG The Fully Coherent Expansion (Remote Translation) Chiplet type connects with a host system to provide application specific acceleration. The chiplet might include an interface to external memory. It is intended for system specific expansion for complex Fully Coherent devices where the trust boundary for the host is at the chiplet interface, for example, when integrating third-party solutions. However, the chiplet is still required to communicate with physical addressing to support generic integration with coherent protocols.

Remote address translation operates as follows:

1. The device in the chiplet requests an address translation from an MMU in the host chiplet, and stores the responses locally.
2. The device applies the received address translation to transactions that it initiates. The transactions are presented to the host chiplet on the chiplet interface with physical addresses.
3. Invalidations of address translations are fed back to the device, so that it discards the locally stored translation information.

This chiplet uses:

- A configuration interface to program the device.
- An interface for requesting address translations.
- An interface for sending MSI interrupts.
- A DMA protocol for data transfer.

A simplified example of a Fully Coherent Expansion (Remote Translation) Chiplet is depicted in Figure 4.9.

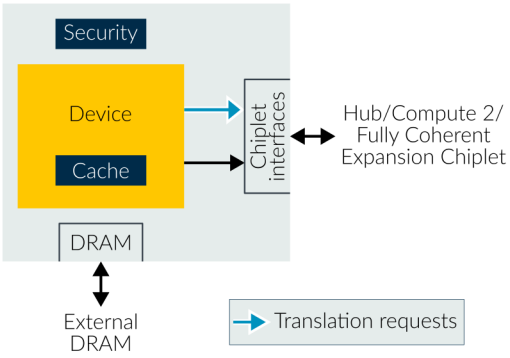

Figure 4.9: Fully Coherent Expansion (Remote Translation) Chiplet Example

A Fully Coherent Expansion (Remote Translation) Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.5.1 Fully Coherent Expansion (Remote Translation) Chiplet Requirements

These requirements are necessary to the definition of the Fully Coherent Expansion (Remote Translation) Chiplet type.

4.5.1.1 Chiplet-to-Chiplet Interfaces

R_{KGJQZ} A Fully Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or more of either:

- A Hub Chiplet
- A Compute 2 Chiplet

(LEVEL-0)

I_{FRDFT} See Fully Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

4.5.1.2 Application PE

R_{KSDBW} A Fully Coherent Expansion (Remote Translation) Chiplet does not contain any Application PEs.

(LEVEL-0)

4.5.1.3 Fully Coherent Device

R_{MHJBP} A Fully Coherent Expansion (Remote Translation) Chiplet contains one or more I/O coherent devices.

(LEVEL-0)

4.5.1.4 I/O Coherent Device

R_{GGJVW} A Fully Coherent Expansion (Remote Translation) Chiplet contains zero or more I/O coherent devices.

(LEVEL-0)

4.5.1.5 I/O Coherent I/O Devices

R_{WDGDV} A Fully Coherent Expansion (Remote Translation) Chiplet contains zero or more I/O coherent I/O devices.

(LEVEL-0)

4.5.1.6 Non-Coherent I/O Device

R_{KLQNT} A Fully Coherent Expansion (Remote Translation) Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.5.1.7 System Memory

R_{TFZGY} A Fully Coherent Expansion (Remote Translation) Chiplet might contain Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{VHJYG} A Fully Coherent Expansion (Remote Translation) Chiplet contains one or more fully coherent agents.

(LEVEL-0)

R_{SVMSC} A Fully Coherent Expansion (Remote Translation) Chiplet might contain caches that support coherency protocol accesses by coherent agents in other chiplets in the system.

(LEVEL-0)

R_{FFDXH} System software accesses memory mapped configuration of devices in a Fully Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

4.5.1.7.1 MMU Location

I_{TJMMT} Agents in a Fully Coherent Expansion (Remote Translation) Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- This MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_{CDQVL} Fully coherent, I/O coherent or non-coherent agents in a Fully Coherent Expansion (Remote Translation) Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is not contained in the chiplet.

(LEVEL-0)

I_{KPMRN} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{HCGBH} A Fully Coherent Expansion (Remote Translation) Chiplet is not within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{KYJPH} Where a Fully Coherent Expansion (Remote Translation) Chiplet is directly connected to a chiplet that is within the system physical addressing trust boundary, that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_{KTTXB} Any agents in a Fully Coherent Expansion (Remote Translation) Chiplet that request address translation information do so from an Arm SMMU [9]. That SMMU is not in the chiplet.

(LEVEL-0)

R_{BJTHM} Memory accesses that are initiated by agents in a Fully Coherent Expansion (Remote Translation) Chiplet and are received by a chiplet that is within the system physical addressing trust boundary are subjected to memory protection checks by an MMU in the receiving chiplet.

(LEVEL-0)

4.5.1.8 Interrupt Handling

I_{LKYKB} This section concerns interrupts that are targeted at application PEs running the main operating system.

Components in a Fully Coherent Expansion (Remote Translation) Chiplet can generate interrupts. These interrupts are forwarded as MSIs.

R_{PJVLK} A Fully Coherent Expansion (Remote Translation) Chiplet transmits interrupts as MSIs to a GIC ITS in a connected chiplet.

(LEVEL-0)

4.5.1.9 Security

R_{JBRRP} A Fully Coherent Expansion (Remote Translation) Chiplet is wholly outside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{VKWXP} Where a Fully Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] the chiplet might contain a component that can generate or is a target for Realm memory transactions. Examples of such components include a device that enables RME-DA/RME-CDA [10], and a Memory Protection Engine (MPE).

R_{PDQFV} Where a Fully Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and contains a component that can generate or is a target for Realm memory transactions, the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{LMJFS} Where a Fully Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and does not contain a component that can generate or is a target for Realm memory transactions, the chiplet is wholly outside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{XTTWV} Where a Fully Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{MBSCP} A Fully Coherent Expansion (Remote Translation) Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{XVZTS} A trusted security agent in a Fully Coherent Expansion (Remote Translation) Chiplet is a secondary trusted security agent.

(LEVEL-0)

4.5.1.10 System Counter

I_{FMYVX} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a Fully Coherent Expansion (Remote Translation) Chiplet might contain a system counter that provides the timestamp value.

R_{PYTWQ} Where a Fully Coherent Expansion (Remote Translation) Chiplet contains a source of CoreSight trace data, it contains a system counter.

(LEVEL-0)

R_{JCTLV} A system counter in a Fully Coherent Expansion (Remote Translation) Chiplet is a secondary system counter.

(LEVEL-0)

4.5.1.11 Debug

I_{NPJVH} A Fully Coherent Expansion (Remote Translation) Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{JSDXW} The debug components contained in a Fully Coherent Expansion (Remote Translation) Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{WXYFJ} A Fully Coherent Expansion (Remote Translation) Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.5.1.12 Trace

I_{JKTHN} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{LNNVR} Where a Fully Coherent Expansion (Remote Translation) Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the Fully Coherent Expansion (Remote Translation) Chiplet or in another chiplet in the system.

(LEVEL-0)

I_{LVTZG} A Fully Coherent Expansion (Remote Translation) Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the Fully Coherent Expansion (Remote Translation) Chiplet.

R_{MKBNF} Where a Fully Coherent Expansion (Remote Translation) Chiplet is interfaced with a chiplet that contains a source of CoreSight trace data, there is a capability to write the trace data collected from components in chiplets that are directly connected to a Fully Coherent Expansion (Remote Translation) Chiplet to the system main memory of a Fully Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

4.5.1.13 System

I_{LKJSJ} A unique identifier is necessary for a Fully Coherent Expansion (Remote Translation) Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{XBJSZ} A Fully Coherent Expansion (Remote Translation) Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

4.6 I/O Coherent Expansion (Translated) Chiplet

INKFQB

The I/O Coherent Expansion (Translated) Chiplet type connects with a host system to provide application specific acceleration and I/O coherent I/O. The chiplet might include interfaces to external DRAM for private use by the application specific acceleration. The chiplet might include interfaces with other I/O coherent or non-coherent chiplet types.

The device(s) access coherent system memory using physical addresses (PA), generated using translation in a memory management unit (MMU) in the chiplet.

Simplified examples of a I/O Coherent Expansion (Translated) Chiplet are depicted in Figure 4.10.

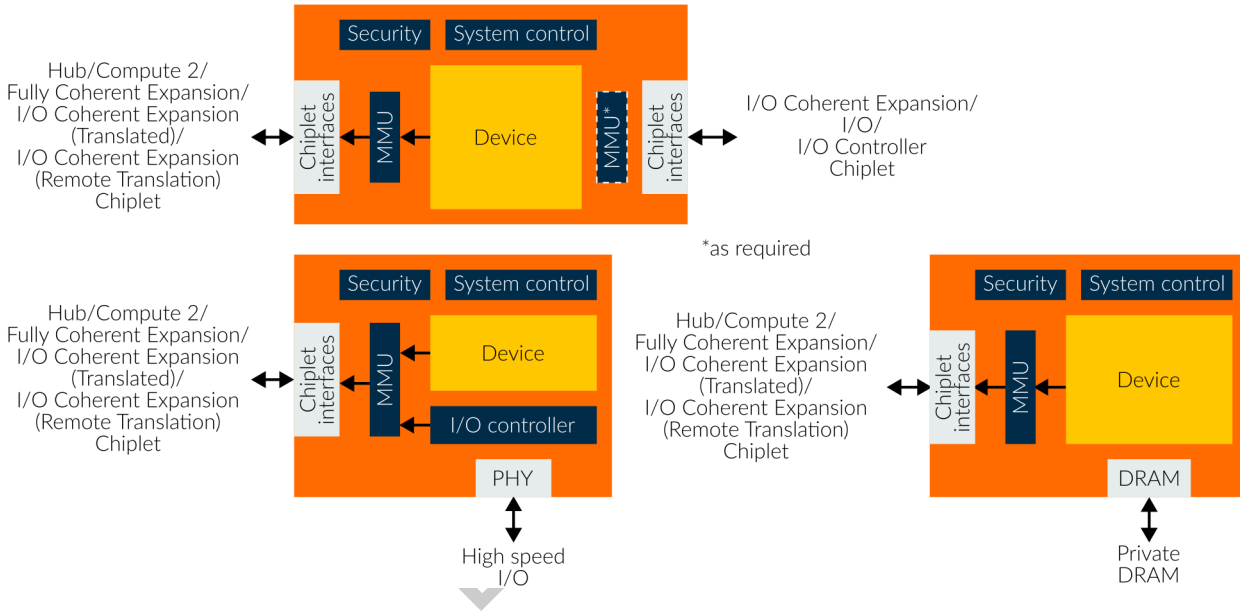

Figure 4.10: I/O Coherent Expansion (Translated) Chiplet Examples

A I/O Coherent Expansion (Translated) Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.6.1 I/O Coherent Expansion (Translated) Chiplet Requirements

These requirements are necessary to the definition of the I/O Coherent Expansion (Translated) Chiplet type.

4.6.1.1 Chiplet-to-Chiplet Interfaces

R_{YTTPM} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more of either:

- A Hub Chiplet.
- A Compute 2 Chiplet.

(LEVEL-0)

R_{RWJXM} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.
(LEVEL-0)

R_{NJQMS} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.

(LEVEL-0)

R_{FHVQW} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more I/O Coherent Expansion (Untranslated) Chiplets.

(LEVEL-0)

R_{XPQFB} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more I/O Coherent Expansion (Remote Translation) Chiplets.

(LEVEL-0)

R_{DFJJB} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more I/O Chiplets.
(LEVEL-0)

R_{LRYYF} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or more I/O Controller Chiplets.
(LEVEL-0)

I_{MTRFG} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

I_{HKZRJ} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{LYJSR} See I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements.

I_{FNWCB} See I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

I_{SGTPD} See I/O Chiplet Interface Requirements.

I_{DBYZT} See I/O Controller Chiplet Interface Requirements.

4.6.1.2 Application PE

R_{NKBDT} A I/O Coherent Expansion (Translated) Chiplet does not contain any Application PEs.
(LEVEL-0)

4.6.1.3 Fully Coherent Device

R_{GPNPF} A I/O Coherent Expansion (Translated) Chiplet does not contain any fully coherent devices.
(LEVEL-0)

4.6.1.4 I/O Coherent Device

R_{JKNBM} A I/O Coherent Expansion (Translated) Chiplet contains one or more I/O coherent devices.
(LEVEL-0)

4.6.1.5 I/O Coherent I/O Device

R_{YLRSX} A I/O Coherent Expansion (Translated) Chiplet contains zero or more I/O coherent I/O devices.
(LEVEL-0)

4.6.1.6 Non-Coherent I/O Device

R_{TXQKB} A I/O Coherent Expansion (Translated) Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.6.1.7 System Memory

R_{MNCCC} A I/O Coherent Expansion (Translated) Chiplet might contain Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{XCBFQ} A I/O Coherent Expansion (Translated) Chiplet does not contain any fully coherent agents.

(LEVEL-0)

R_{BGYTR} A I/O Coherent Expansion (Translated) Chiplet does not contain any caches that support coherency protocol accesses by coherent agents in other chiplets in the system.

(LEVEL-0)

4.6.1.7.1 MMU Location

I_{LWGQZ} Agents in a I/O Coherent Expansion (Translated) Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_{YSCVM} I/O coherent or non-coherent agents in a I/O Coherent Expansion (Translated) Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is contained in the chiplet.

(LEVEL-0)

I_{WNYJY} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{XDDMZ} A I/O Coherent Expansion (Translated) Chiplet is within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{XWPCG} Where a I/O Coherent Expansion (Translated) Chiplet is directly connected to a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries), that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_{DGKNW} Memory accesses that are initiated by agents in a chiplet that is not within the system physical addressing trust boundary (see Trust Boundaries) and are received by a I/O Coherent Expansion (Translated) Chiplet are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the I/O Coherent Expansion (Translated) Chiplet coincides with crossing to within the system physical addressing trust boundary, this MMU is in the I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

4.6.1.8 Interrupt Handling

I_{HFHBL} This section concerns interrupts that are targeted at application PEs running the main operating system.

Interrupts in a I/O Coherent Expansion (Translated) Chiplet are handled according to the *Arm Generic Interrupt Controller (GIC) Architecture* [4].

A I/O Coherent Expansion (Translated) Chiplet can be a source of interrupts, or convey interrupts from other chiplets. Therefore, the chiplet can:

- Generate an interrupt internally, with the destination being a PE in another chiplet.
- Receive an interrupt from another chiplet, with the destination being a PE in another chiplet.

The components in a I/O Coherent Expansion (Translated) Chiplet can generate interrupts, or interrupts can be received from other chiplets that do not have a GIC distributor (GICD) component. These interrupts are either :

- If the chiplet has a GICD, received at the I/O Coherent Expansion (Translated) Chiplet GICD and then forwarded to the GICD in the chiplet that contains the target PE.
- If the chiplet does not have a GICD, forwarded to a connected chiplets GICD or ITS, dependent on interrupt type.

R_{VCSNX} A I/O Coherent Expansion (Translated) Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

U_{SPBXD} The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_{HMGLJ} A I/O Coherent Expansion (Translated) Chiplet can contain a GIC Distributor (GICD).

(LEVEL-0)

R_{ZHDSR} When a I/O Coherent Expansion (Translated) Chiplet contains a GICD, the GICD receives interrupts from within the same chiplet.

(LEVEL-0)

R_{LPFHQ} When a I/O Coherent Expansion (Translated) Chiplet contains a GICD, the GICD can receive interrupts from other chiplets in the system, that do not contain a GICD.

(LEVEL-0)

R_{SXYNL} When a I/O Coherent Expansion (Translated) Chiplet contains a GICD, where the target PE of an interrupt is in another chiplet, the GICD forwards the interrupt to the GICD in the chiplet of the target PE.

(LEVEL-0)

R_{MPXKH} When a I/O Coherent Expansion (Translated) Chiplet contains a GICD, if the I/O Coherent Expansion (Translated) Chiplet receives MSIs from connected chiplets, it contains an Interrupt Translation Service (ITS) component.

(LEVEL-0)

I_{WZBKG} The GIC architecture [4] specifies a *Multiprocessor Affinity Register* MPIDR_EL1 that provides a PE identification mechanism for the purpose of scheduling interrupts. Affinity routing is a hierarchical address-based scheme for identifying specific PEs for interrupt routing. The architecture does not specify at what point the values of MPIDR_EL1 are set. The routing decisions made by the GIC using MPIDR_EL1 are made by software.

R_{TKMWG} When a I/O Coherent Expansion (Translated) Chiplet contains a GICD, the method used to set the value of the *Multiprocessor Affinity Register* MPIDR_EL1 in a GIC in a I/O Coherent Expansion (Translated) Chiplet does not conflict with the values of the MPIDR_EL1 registers of other GIC components in other chiplets in the system.

(LEVEL-0)

R_{MGQGR} When a I/O Coherent Expansion (Translated) Chiplet does not contain a GICD, it transmits LPIs as MSIs to a GIC ITS in a connected chiplet.

(LEVEL-0)

R_{YXVZS} When a I/O Coherent Expansion (Translated) Chiplet does not contain a GICD, it transmits SPI interrupts as IMPLEMENTATION DEFINED messages to a GICD in a connected chiplet.

(LEVEL-0)

4.6.1.9 Chiplet System Control

R_{TWRHN} a I/O Coherent Expansion (Translated) Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R_{ZHTQD} The system controller in a I/O Coherent Expansion (Translated) Chiplet is a secondary system controller.

(LEVEL-0)

4.6.1.10 Security

R_{BVSBW} A I/O Coherent Expansion (Translated) Chiplet is wholly inside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{TKFLQ} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] the chiplet might contain a component that can generate or is a target for Realm memory transactions. Examples of such components include a device that enables RME-DA/RME-CDA [10], and a Memory Protection Engine (MPE).

R_{PRMCJ} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and contains a component that can generate or is a target for Realm memory transactions, the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{YJYLG} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] the chiplet might be interfaced with another chiplet that can generate or is a target for Realm memory transactions. The chiplet facilitates the transport of the Realm memory transactions to and from that other chiplet.

R_{XJDKV} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and facilitates the transport of Realm memory transactions to and from another chiplet that it might be interfaced with, the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{BXZMH} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and neither contains a component that can generate or is a target for Realm memory transactions nor facilitates the transport of Realm memory transactions to and from another chiplet that it might be interfaced with, the chiplet is wholly outside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{LTWFM} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the MMU it contains supports the Granule Protection Check (GPC) and makes Root accesses to the Granule Protection Table (GPT).

R_{KKYCW} Where a I/O Coherent Expansion (Translated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly inside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{LCWDK} A I/O Coherent Expansion (Translated) Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{JTTCQ} The trusted security agents in a I/O Coherent Expansion (Translated) Chiplet are secondary trusted security agents.

(LEVEL-0)

4.6.1.11 System Counter

I_{DQZRH} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a I/O Coherent Expansion (Translated) Chiplet might contain a system counter that provides the timestamp value.

R_{QLSDY} Where a I/O Coherent Expansion (Translated) Chiplet contains a source of CoreSight trace data, it contains a system counter.

(LEVEL-0)

R_{TWVMQ} A system counter in a I/O Coherent Expansion (Translated) Chiplet is a secondary system counter.

(LEVEL-0)

4.6.1.12 Debug

I_{QDJFZ} A I/O Coherent Expansion (Translated) Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{ZKGSN} The debug components contained in a I/O Coherent Expansion (Translated) Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{JYTSS} A I/O Coherent Expansion (Translated) Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.6.1.13 Trace

I_{HYJRX} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{KKPCM} Where a I/O Coherent Expansion (Translated) Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the I/O Coherent Expansion (Translated) Chiplet or in another chiplet in the system.

(LEVEL-0)

I_{YBZWR} A I/O Coherent Expansion (Translated) Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the I/O Coherent Expansion (Translated) Chiplet or in another chiplet in the system.

R_{XPJTN} Where a I/O Coherent Expansion (Translated) Chiplet is interfaced with a chiplet that contains a source of CoreSight trace data, there is a capability to write the trace data collected from components in chiplets that are directly connected to a I/O Coherent Expansion (Translated) Chiplet to the system main memory of a I/O Coherent Expansion (Translated) Chiplet or of other chiplets in the system.

(LEVEL-0)

4.6.1.14 System

I_{TCKLP} A unique identifier is necessary for a I/O Coherent Expansion (Translated) Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{DDQVP} A I/O Coherent Expansion (Translated) Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

4.7 I/O Coherent Expansion (Untranslated) Chiplet

Note

Arm is requesting feedback on the need for this chiplet type. Arm currently recommends using the I/O Coherent Expansion (Translated) Chiplet for flexibility and IP support.

I_{TLKDT} The I/O Coherent Expansion (Untranslated) Chiplet type connects with a host system to provide application specific acceleration and I/O coherent I/O. The chiplet might include an interface to external DRAM for private use by the application specific acceleration. The chiplet might include interfaces with other I/O coherent or non-coherent chiplet types.

The chiplet provides system specific expansion for functions such as existing I/O coherent AMBA devices while adding only minimal additional complexity to the chiplet. Functions such as system control, interrupt handling and address translation are managed in attached chiplets.

Simplified examples of a I/O Coherent Expansion (Untranslated) Chiplet are depicted in Figure 4.11.

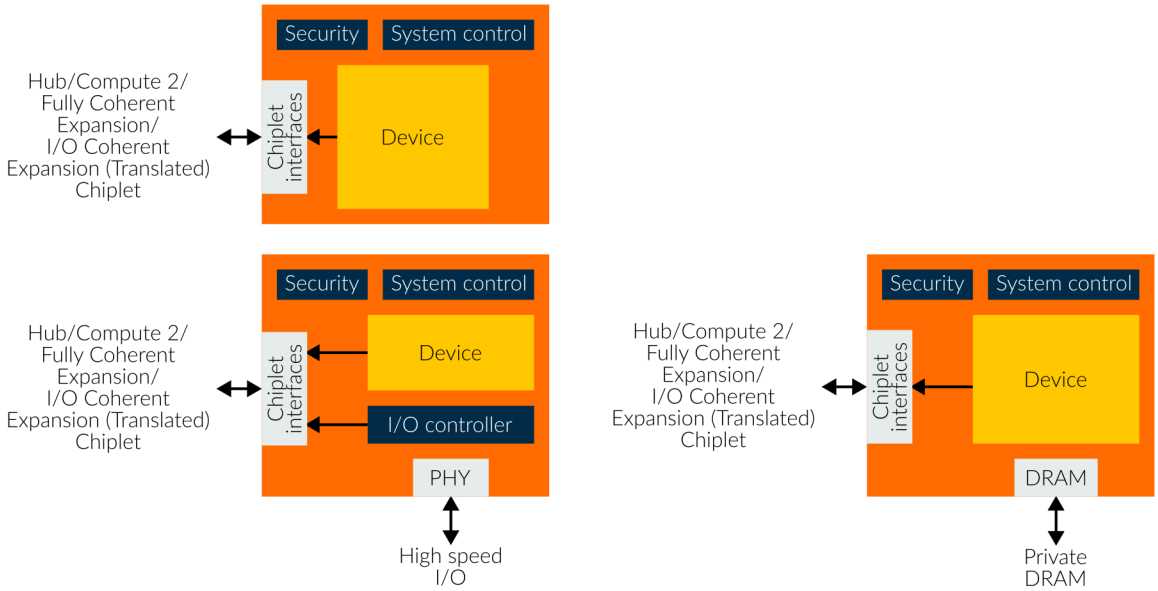

Figure 4.11: I/O Coherent Expansion (Untranslated) Chiplet Examples

A I/O Coherent Expansion (Untranslated) Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.7.1 I/O Coherent Expansion (Untranslated) Chiplet Requirements

These requirements are necessary to the definition of the I/O Coherent Expansion (Untranslated) Chiplet type.

4.7.1.1 Chiplet-to-Chiplet Interfaces

R_{YWWFY} A I/O Coherent Expansion (Untranslated) Chiplet is interfaced with zero or more of either:

- A Hub Chiplet.
- A Compute 2 Chiplet.

(LEVEL-0)

R_{WFLXB} A I/O Coherent Expansion (Untranslated) Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.

(LEVEL-0)

R_{JHQPD} A I/O Coherent Expansion (Untranslated) Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.

(LEVEL-0)

I_{DWDHZ} See I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements.

I_{JKFCD} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{NCBJR} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

4.7.1.2 Application PE

R_{BNCHJ} A I/O Coherent Expansion (Untranslated) Chiplet does not contain any Application PEs.

(LEVEL-0)

4.7.1.3 Fully Coherent Device

R_{DGKSX} A I/O Coherent Expansion (Untranslated) Chiplet does not contain any fully coherent devices.

(LEVEL-0)

4.7.1.4 I/O Coherent Device

R_{HKCGS} A I/O Coherent Expansion (Untranslated) Chiplet contains one or more I/O coherent devices.

(LEVEL-0)

4.7.1.5 I/O Coherent I/O Device

R_{YHDSL} A I/O Coherent Expansion (Untranslated) Chiplet contains zero or more I/O coherent I/O devices.

(LEVEL-0)

4.7.1.6 Non-Coherent I/O Device

R_{DPLFM} A I/O Coherent Expansion (Untranslated) Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.7.1.7 System Memory

R_{NDGMQ} A I/O Coherent Expansion (Untranslated) Chiplet does not contain system main memory, either integrated or directly connected.

(LEVEL-0)

R_{RGMBN} A I/O Coherent Expansion (Untranslated) Chiplet does not contain any fully coherent agents.

(LEVEL-0)

R_{NPHFD} A I/O Coherent Expansion (Untranslated) Chiplet does not contain any caches that support coherency protocol accesses by coherent agents in other chiplets in the system.

(LEVEL-0)

R_{KFPMT} System software accesses memory mapped configuration of devices in a I/O Coherent Expansion (Untranslated) Chiplet.

(LEVEL-0)

4.7.1.7.1 MMU Location

I_{JPSMF} Agents in a I/O Coherent Expansion (Untranslated) Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_{VSTXF} I/O coherent or non-coherent agents in a I/O Coherent Expansion (Untranslated) Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is not contained in the chiplet.

(LEVEL-0)

I_{RKDZF} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{KWSDM} A I/O Coherent Expansion (Untranslated) Chiplet is not within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{WFVTN} Where a I/O Coherent Expansion (Untranslated) Chiplet is directly connected to a chiplet that is within the system physical addressing trust boundary, that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_{GXRYN} Memory accesses that are initiated by agents in a I/O Coherent Expansion (Untranslated) Chiplet and are received by a chiplet that is within the system physical addressing trust boundary (see Trust Boundaries) are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the receiving chiplet coincides with crossing to within the system physical addressing trust boundary, this MMU is in the receiving chiplet.

(LEVEL-0)

4.7.1.8 Interrupt Handling

I_{JCLGP} This section concerns interrupts that are targeted at application PEs running the main operating system.

The components on a I/O Coherent Expansion (Untranslated) Chiplet can generate interrupts.

These interrupts are forwarded either:

- For Locality-Specific Interrupts (LPI), as an MSI to a GIC Interrupt Translation Service (ITS) component in an attached chiplet.
- For Shared Peripheral Interrupts (SPI), as an IMPLEMENTATION DEFINED message to a GIC Distributor (GICD) component.

R_{YWQVN} A I/O Coherent Expansion (Untranslated) Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

U_{GFBMH} The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_{YHQVD} A I/O Coherent Expansion (Untranslated) Chiplet transmits LPIs as MSIs to a GIC ITS in a connected chiplet.

(LEVEL-0)

R_{PJYKG} A I/O Coherent Expansion (Untranslated) Chiplet transmits SPIs as IMPLEMENTATION DEFINED messages to a GICD on a connected compute or hub chiplet.

(LEVEL-0)

4.7.1.9 Chiplet System Control

R_{KMTCS} A I/O Coherent Expansion (Untranslated) Chiplet contains a system control agent for reset and power mode control of the chiplet.

(LEVEL-0)

R_{JMJYS} The PMIC associated with a I/O Coherent Expansion (Untranslated) Chiplet is controlled by the primary system controller.

(LEVEL-0)

4.7.1.10 Security

R_{KXJWP} A I/O Coherent Expansion (Untranslated) Chiplet is wholly outside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{JNHJX} Where a I/O Coherent Expansion (Untranslated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] the chiplet might contain a component that can generate or is a target for Realm memory transactions. Examples of such components include a device that enables RME-DA/RME-CDA [10], and a Memory Protection Engine (MPE).

R_{TGDQV} Where a I/O Coherent Expansion (Untranslated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and contains a component that can generate or is a target for Realm memory transactions, the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{BVFLF} Where a I/O Coherent Expansion (Untranslated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and does not contain a component that can generate or is a target for Realm memory transactions, the chiplet is wholly outside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{LDHBH} Where a I/O Coherent Expansion (Untranslated) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{HHJCC} A I/O Coherent Expansion (Untranslated) Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{MGPTM} A trusted security agent in a I/O Coherent Expansion (Untranslated) Chiplet is a secondary trusted security agent.

(LEVEL-0)

4.7.1.11 System Counter

I_{DBYJT} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a I/O Coherent Expansion (Untranslated) Chiplet might contain a system counter that provides the timestamp value.

R_{YQFDX} Where a I/O Coherent Expansion (Untranslated) Chiplet contains a source of CoreSight trace data, it contains a system counter.

(LEVEL-0)

R_{LXWHF} A system counter in a I/O Coherent Expansion (Untranslated) Chiplet is a secondary system counter.

(LEVEL-0)

4.7.1.12 Debug

I_{ZQBPJ} A I/O Coherent Expansion (Untranslated) Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{PJLCZ} The debug components contained in a I/O Coherent Expansion (Untranslated) Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{DXFGF} A I/O Coherent Expansion (Untranslated) Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.7.1.13 Trace

I_{CNDHX} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{JZJYR} Where a I/O Coherent Expansion (Untranslated) Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in another chiplet in the system.

(LEVEL-0)

4.7.1.14 System

I_{VFMRV} A unique identifier is necessary for a I/O Coherent Expansion (Untranslated) Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{WSWRW} A I/O Coherent Expansion (Untranslated) Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

4.8 I/O Coherent Expansion (Remote Translation) Chiplet

Note

Arm is requesting feedback on the need for this chiplet type. Arm currently recommends using the I/O Coherent Expansion (Translated) Chiplet for flexibility and IP support.

IWJTFD

The I/O Coherent Expansion (Remote Translation) Chiplet type connects with a host system to provide application specific acceleration and I/O coherent I/O. The chiplet might include an interface to external memory for private use by the application specific acceleration.

It is intended for system specific expansion for complex I/O Coherent devices where the trust boundary for the host is at the chiplet interface, for example, when integrating third-party solutions. However, the chiplet is still required to communicate with physical addressing to support generic integration with coherent protocols.

Remote address translation operates as follows:

1. The device in the chiplet requests an address translation from an MMU in the host chiplet, and stores the responses locally.
2. The device applies the received address translation to transactions that it initiates. The transactions are presented to the host chiplet on the chiplet interface with physical addresses.
3. Invalidations of address translations are fed back to the device, so that it discards the locally stored translation information.

This chiplet uses:

- A configuration interface to program the device.
- An interface for requesting address translations.
- An interface for sending MSI interrupts.
- A DMA protocol for data transfer.

Simplified examples of a I/O Coherent Expansion (Remote Translation) Chiplet are depicted in Figure 4.12.

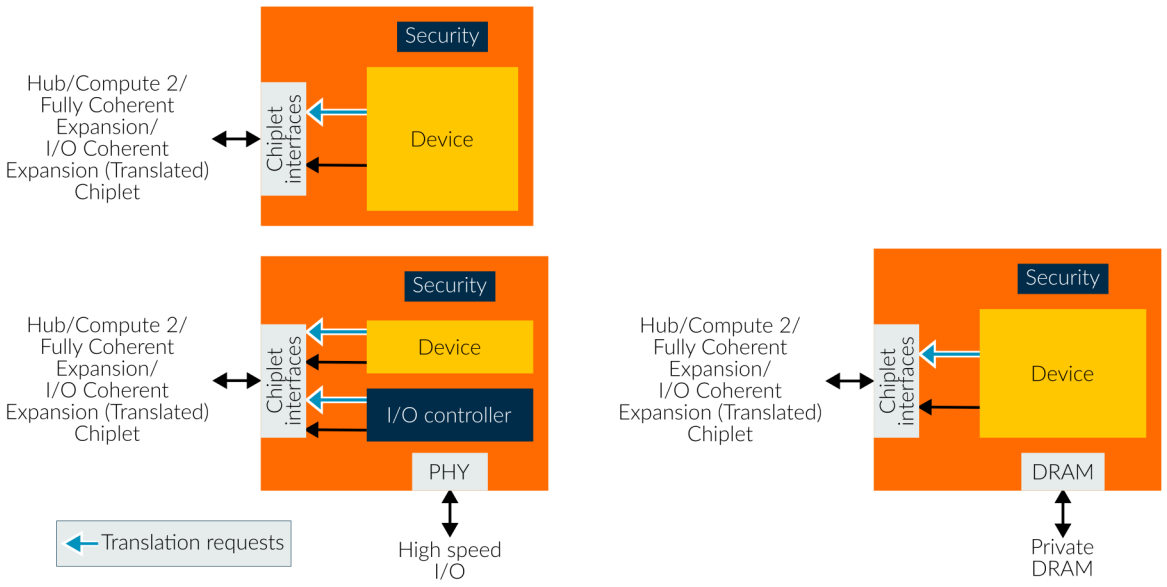

Figure 4.12: I/O Coherent Expansion (Remote Translation) Chiplet Examples

A I/O Coherent Expansion (Remote Translation) Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.8.1 I/O Coherent Expansion (Remote Translation) Chiplet Requirements

These requirements are necessary to the definition of the I/O Coherent Expansion (Remote Translation) Chiplet type.

4.8.1.1 Chiplet-to-Chiplet Interfaces

R_{MCTLV} A I/O Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or more of either:

- A Hub Chiplet
- A Compute 2 Chiplet

(LEVEL-0)

I_{KMCHP} See I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements.

4.8.1.2 Application PE

R_{DFKTT} A I/O Coherent Expansion (Remote Translation) Chiplet does not contain any Application PEs.

(LEVEL-0)

4.8.1.3 Fully Coherent Device

R_{XKHNG} A I/O Coherent Expansion (Remote Translation) Chiplet does not contain any fully coherent devices.

(LEVEL-0)

4.8.1.4 I/O Coherent Device

R_{RXVFP} A I/O Coherent Expansion (Remote Translation) Chiplet contains one or more I/O coherent devices.

(LEVEL-0)

4.8.1.5 I/O Coherent I/O Device

R_{SJJRK} A I/O Coherent Expansion (Remote Translation) Chiplet contains zero or more I/O coherent I/O devices.

(LEVEL-0)

4.8.1.6 Non-Coherent I/O Device

R_{VGYHF} A I/O Coherent Expansion (Remote Translation) Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.8.1.7 System Memory

R_{SYMRV} A I/O Coherent Expansion (Remote Translation) Chiplet might contain Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{MGPMV} A I/O Coherent Expansion (Remote Translation) Chiplet does not contain any fully coherent agents.

(LEVEL-0)

R_{MJDDS} A I/O Coherent Expansion (Remote Translation) Chiplet does not contain any caches that support coherency protocol accesses by coherent agents in other chiplets in the system.

(LEVEL-0)

R_{VLFCJ} System software accesses memory mapped configuration of devices in a I/O Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

4.8.1.7.1 MMU Location

I_{CKFKS} Agents in a I/O Coherent Expansion (Remote Translation) Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- This MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_{XDGPZ} I/O coherent or non-coherent agents in a I/O Coherent Expansion (Remote Translation) Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is not contained in the chiplet.

(LEVEL-0)

I_{HGZST} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{TXXQS} A I/O Coherent Expansion (Remote Translation) Chiplet is not within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{KCVLT} Where a I/O Coherent Expansion (Remote Translation) Chiplet is directly connected to a chiplet that is within the system physical addressing trust boundary, that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_{FJJKL} Any agents in a I/O Coherent Expansion (Remote Translation) Chiplet that request address translation information do so from an Arm SMMU [9]. That SMMU is not in the chiplet.

(LEVEL-0)

R_{ZNCPC} Memory accesses that are initiated by agents in a I/O Coherent Expansion (Remote Translation) Chiplet and are received by a chiplet that is within the system physical addressing trust boundary are subjected to memory protection checks by an MMU in the receiving chiplet.

(LEVEL-0)

4.8.1.8 Interrupt Handling

I_{JQSTB} This section concerns interrupts that are targeted at application PEs running the main operating system.

The components on a I/O Coherent Expansion (Remote Translation) Chiplet can generate interrupts. These interrupts are forwarded as MSIs.

R_{SSKPN} A I/O Coherent Expansion (Remote Translation) Chiplet transmits interrupts as MSIs to a GIC ITS in a connected chiplet.

(LEVEL-0)

4.8.1.9 Security

R_{VPTCL} A I/O Coherent Expansion (Remote Translation) Chiplet is wholly outside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

I_{GNCRQ} Where a I/O Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] the chiplet might contain a component that can generate or is a target for Realm memory transactions. Examples of such components include a device that enables RME-DA/RME-CDA [10], and a Memory Protection Engine (MPE).

R_{CKXDG} Where a I/O Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and contains a component that can generate or is a target for Realm memory transactions, the chiplet is wholly inside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{LCWHG} Where a I/O Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5] and does not contain a component that can generate or is a target for Realm memory transactions, the chiplet is wholly outside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{DDZZJ} Where a I/O Coherent Expansion (Remote Translation) Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{VVWXF} A I/O Coherent Expansion (Remote Translation) Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{MTJGB} A trusted security agent in a I/O Coherent Expansion (Remote Translation) Chiplet is a secondary trusted security agent.

(LEVEL-0)

4.8.1.10 System Counter

I_{PBVZN} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a I/O Coherent Expansion (Remote Translation) Chiplet might contain a system counter that provides the timestamp value.

R_{GMXLY} Where a I/O Coherent Expansion (Remote Translation) Chiplet contains a source of CoreSight trace data, it contains a system counter.

(LEVEL-0)

R_{JWTCB} A system counter in a I/O Coherent Expansion (Remote Translation) Chiplet is a secondary system counter.

(LEVEL-0)

4.8.1.11 Debug

I_{JGNHX} A I/O Coherent Expansion (Remote Translation) Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{WXTJK} The debug components contained in a I/O Coherent Expansion (Remote Translation) Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{MKFTB} A I/O Coherent Expansion (Remote Translation) Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.8.1.12 Trace

I_{JJRKN} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{PFTPX} Where a I/O Coherent Expansion (Remote Translation) Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in the I/O Coherent Expansion (Remote Translation) Chiplet or in another chiplet in the system.

(LEVEL-0)

I_{RGYMQ} A I/O Coherent Expansion (Remote Translation) Chiplet might be interfaced with a chiplet that contains a source of CoreSight trace data and facilitates the transfer of that trace data to system main memory. That memory might be in the I/O Coherent Expansion (Remote Translation) Chiplet.

R_{HNKXF} Where a I/O Coherent Expansion (Remote Translation) Chiplet is interfaced with a chiplet that contains a source of CoreSight trace data, there is a capability to write the trace data collected from components in chiplets that are directly connected to a I/O Coherent Expansion (Remote Translation) Chiplet to the system main memory of a I/O Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

4.8.1.13 System

I_{QRVVW} A unique identifier is necessary for a I/O Coherent Expansion (Remote Translation) Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{DZVCK} A I/O Coherent Expansion (Remote Translation) Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

4.9 I/O Chiplet

IGXMMN The I/O Chiplet type connects with a host system to provide I/O coherent I/O.

It is intended as system specific expansion for I/O coherent I/O devices while adding only minimal additional complexity to the chiplet. Functions such as system control, interrupt handling and address translation are managed in attached chiplets.

A simplified example of a I/O Chiplet is depicted in Figure 4.13.

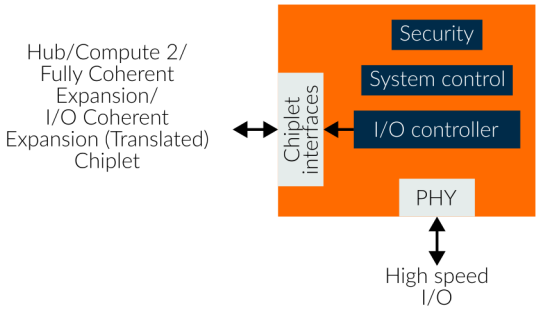

Figure 4.13: I/O Chiplet Example

A I/O Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.9.1 I/O Chiplet Requirements

These requirements are necessary to the definition of the I/O Chiplet type.

4.9.1.1 Chiplet-to-Chiplet Interfaces

R_{FKNWS} A I/O Chiplet is interfaced with zero or more of either:

- A Hub Chiplet.
- A Compute 2 Chiplet.

(LEVEL-0)

R_{CFTXC} A I/O Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.

(LEVEL-0)

R_{FHZFP} A I/O Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.

(LEVEL-0)

I_{KWXNR} See I/O Chiplet Interface Requirements.

I_{DVHCD} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{NJNHQ} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

4.9.1.2 Application PE

R_{ZQPNC} A I/O Chiplet does not contain any Application PEs.

(LEVEL-0)

4.9.1.3 Fully Coherent Device

R_{JHMFZ} A I/O Chiplet does not contain any fully coherent devices.

(LEVEL-0)

4.9.1.4 I/O Coherent Device

R_{KWXTX} A I/O Chiplet does not contain any I/O coherent devices.

(LEVEL-0)

4.9.1.5 I/O Coherent I/O Device

R_{CPTMG} A I/O Chiplet contains one or more I/O coherent I/O devices.

(LEVEL-0)

4.9.1.6 Non-Coherent I/O Device

R_{ZMHCY} A I/O Chiplet contains zero or more non-coherent I/O devices.

(LEVEL-0)

4.9.1.7 System Memory

R_{KHZVP} A I/O Chiplet might contain Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{YRKBX} A I/O Chiplet does not contain any fully coherent agents.

(LEVEL-0)

R_{JSKVP} A I/O Chiplet does not contain any caches that support coherency protocol accesses by coherent agents in other chiplets in the system.

(LEVEL-0)

R_{RRCYS} System software accesses memory mapped configuration of devices in a I/O Chiplet.

(LEVEL-0)

4.9.1.7.1 MMU Location

I_{NFQMM} Agents in a I/O Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_{KXVHZ} I/O coherent or non-coherent agents in a I/O Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is not contained in the chiplet.
(LEVEL-0)

I_{PNVXX} The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_{GDTMR} A I/O Chiplet is not within the system physical addressing trust boundary (see Trust Boundaries).
(LEVEL-0)

R_{KCVWN} Where a I/O Chiplet is directly connected to a chiplet that is within the system physical addressing trust boundary, that boundary is located at the interface between the two chiplets.
(LEVEL-0)

R_{TGMWL} Memory accesses that are initiated by agents that are in a I/O Chiplet and are received by a chiplet that is within the system physical addressing trust boundary (see Trust Boundaries) are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the receiving chiplet coincides with crossing to within the system physical addressing trust boundary, this MMU is in the receiving chiplet.
(LEVEL-0)

4.9.1.8 Interrupt Handling

I_{GPSWK} This section concerns interrupts that are targeted at application PEs running the main operating system.

The components in a I/O Chiplet can generate interrupts.

These interrupts are forwarded either:

- For Locality-Specific Interrupts (LPI), as an MSI to a GIC Interrupt Translation Service (ITS) component in an attached chiplet.
- For Shared Peripheral Interrupts (SPI), as an IMPLEMENTATION DEFINED message to a GIC Distributor (GICD) component.

R_{JSWCC} A I/O Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.
(LEVEL-0)

U_{DNDTX} The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_{GDKCJ} A I/O Chiplet transmits LPIs as MSIs to a GIC ITS in a connected chiplet.
(LEVEL-0)

R_{WKBWW} A I/O Chiplet transmits SPI interrupts as IMPLEMENTATION DEFINED messages to a GICD in a connected chiplet.
(LEVEL-0)

4.9.1.9 Chiplet System Control

R_{LCHML} A I/O Chiplet contains a system control agent for reset and power mode control of the chiplet.
(LEVEL-0)

R_{XJFYN} The PMIC associated with a I/O Chiplet is controlled by the primary system controller.
(LEVEL-0)

4.9.1.10 Security

R_{CTTLN} A I/O Chiplet is wholly outside the Secure memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{TKBBC} Where a I/O Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Realm memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{XTWDY} Where a I/O Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Root memory trust boundary (see Trust Boundaries).

(LEVEL-0)

R_{WPDRB} A I/O Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.

(LEVEL-0)

R_{BNDMC} A trusted security agent in a I/O Chiplet is a secondary trusted security agent.

(LEVEL-0)

4.9.1.11 System Counter

I_{WQNGV} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a I/O Chiplet might contain a system counter that provides the timestamp value.

R_{HGNYQ} Where a I/O Chiplet contains a source of CoreSight trace data, it contains a system counter.

(LEVEL-0)

R_{RCNGP} A system counter in a I/O Chiplet is a secondary system counter.

(LEVEL-0)

4.9.1.12 Debug

I_{HBMGP} A I/O Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{YFNFR} The debug components contained in a I/O Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.

(LEVEL-0)

R_{FWSCR} A I/O Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources in other chiplets.

(LEVEL-0)

4.9.1.13 Trace

I_{MPVHG} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{YDSWZ} Where a I/O Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in another chiplet in the system.

(LEVEL-0)

4.9.1.14 System

I_{MQGLT} A unique identifier is necessary for a I/O Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{BXGXT} A I/O Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

4.10 I/O Controller Chiplet

IDGSTG The I/O Controller Chiplet type provides high speed I/O to the system, including PHYs, such as DDR SDRAM, USB, MIPI CSI or HDMI. There are no coherent agents in the chiplet.

A simplified example of a I/O Controller Chiplet is depicted in Figure 4.14.

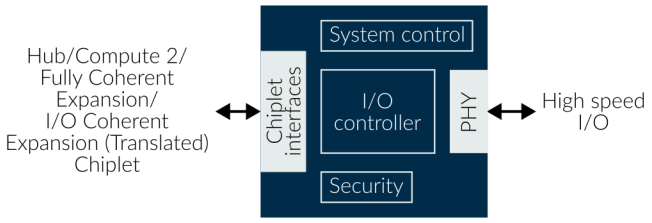

Figure 4.14: I/O Controller Chiplet Example

A I/O Controller Chiplet is a constituent of the following system types:

- Compute & Hub System.
- Compute Tile System.

Refer to the system type definitions for additional system-level specification rules that apply to this chiplet type.

4.10.1 I/O Controller Chiplet Requirements

These requirements are necessary to the definition of the I/O Controller Chiplet type.

4.10.1.1 Chiplet-to-Chiplet Interfaces

R_{FFNKP} A I/O Controller Chiplet is interfaced with zero or more of either:

- A Hub Chiplet.
- A Compute 2 Chiplet.

(LEVEL-0)

R_{RDQYR} A I/O Controller Chiplet is interfaced with zero or more Fully Coherent Expansion Chiplets.

(LEVEL-0)

R_{DSBQD} A I/O Controller Chiplet is interfaced with zero or more I/O Coherent Expansion (Translated) Chiplets.

(LEVEL-0)

I_{CPZPP} See I/O Controller Chiplet Interface Requirements.

I_{CJMRS} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{WCJZH} See I/O Coherent Expansion (Translated) Chiplet Interface Requirements.

4.10.1.2 Application PE

R_{WBVMW} A I/O Controller Chiplet does not contain any Application PEs.

(LEVEL-0)

4.10.1.3 Fully Coherent Device

R_{XWTXV} A I/O Controller Chiplet does not contain any fully coherent devices.

(LEVEL-0)

4.10.1.4 I/O Coherent Device

R_{BNQNG} A I/O Controller Chiplet does not contain any I/O coherent devices.

(LEVEL-0)

4.10.1.5 I/O Coherent I/O Device

R_{JRSRT} A I/O Controller Chiplet does not contain any I/O coherent I/O devices.

(LEVEL-0)

4.10.1.6 Non-Coherent I/O Device

R_{RZGDN} A I/O Controller Chiplet contains one or more non-coherent I/O devices.

(LEVEL-0)

4.10.1.7 System Memory

R_{QSWDY} A I/O Controller Chiplet might contain Normal Cacheable or Normal Non-cacheable system main memory, either integrated or directly connected.

(LEVEL-0)

R_{LZRDM} A I/O Controller Chiplet does not contain any fully coherent agents.

(LEVEL-0)

R_{MWPHD} A I/O Controller Chiplet does not contain any I/O coherent agents.

(LEVEL-0)

R_{DWLLK} A I/O Controller Chiplet does not contain any caches that support coherency protocol accesses by coherent agents in other chiplets in the system.

(LEVEL-0)

4.10.1.7.1 MMU Location

I_RPFRQ Agents in a I/O Controller Chiplet access system addressable memory. All agents in the system access memory through an MMU that provides address translation and memory protection.

- For I/O coherent or non-coherent agents, this MMU follows the *Arm System Memory Management Unit Architecture Specification* [9].

R_CKFDQ I/O coherent or non-coherent agents in a I/O Controller Chiplet access system memory through an MMU as specified by the *Arm System Memory Management Unit Architecture Specification* [9]. That MMU is not contained in the chiplet.

(LEVEL-0)

I_WDQQK The CSA specifies that the system physical addressing trust boundary (see Trust Boundaries) is located at the interface between two chiplets, where one of the two chiplets is within the boundary, and the other is not. The interface between two chiplets is a location that reflects the encapsulation of memory translation and protection functionality.

R_VHQZD A I/O Controller Chiplet is not within the system physical addressing trust boundary (see Trust Boundaries).

(LEVEL-0)

R_NJRSR Where a I/O Controller Chiplet is directly connected to a chiplet that is within the system physical addressing trust boundary, that boundary is located at the interface between the two chiplets.

(LEVEL-0)

R_GXMYX Memory accesses that are initiated by agents in a I/O Controller Chiplet and are received by a chiplet that is within the system physical addressing trust boundary (see Trust Boundaries) are translated to the physical address space (in the case of remote translation, the agent is provided with address translation information) and subjected to memory protection checks by an MMU. Where the receipt of such transactions by the receiving chiplet coincides with crossing to within the system physical addressing trust boundary, this MMU is in the receiving chiplet.

(LEVEL-0)

4.10.1.8 Interrupt Handling

I_ZVJZG This section concerns interrupts that are targeted at application PEs running the main operating system.

The components in a I/O Controller Chiplet can generate interrupts.

All interrupts are handled using the *Arm Generic Interrupt Controller Architecture* [4].

These interrupts are forwarded either:

- For Locality-Specific Interrupts (LPI), as an MSI to a GIC Interrupt Translation Service (ITS) component in an attached chiplet.
- For Shared Peripheral Interrupts (SPI), as an IMPLEMENTATION DEFINED message to a GIC Distributor (GICD) component.

R_VSNSX A I/O Controller Chiplet uses the *Arm Generic Interrupt Controller Architecture* [4] to manage interrupts.

(LEVEL-0)

U_PQKKH The structure and implementation rules given here reflect the HW implementation of the GIC, where the partitioning of components is slightly different from the conceptual split given in the *Arm Generic Interrupt Controller Architecture* [4]. This does not affect the SW architectural view. For more details see [4] and [6].

R_BMYSB A I/O Controller Chiplet transmits LPIs as MSIs to a GIC ITS in a connected chiplet.

(LEVEL-0)

R_DTLYG A I/O Controller Chiplet transmits SPI interrupts as IMPLEMENTATION DEFINED messages to a GICD in a connected chiplet.

(LEVEL-0)

4.10.1.9 Chiplet System Control

R_{KZQVC} A I/O Controller Chiplet contains a system control agent for reset and power mode control of the chiplet.
(LEVEL-0)

R_{PDQHN} The PMIC associated with a I/O Controller Chiplet is controlled by the primary system controller.
(LEVEL-0)

4.10.1.10 Security

R_{KKFWR} A I/O Controller Chiplet is wholly outside the Secure memory trust boundary (see Trust Boundaries).
(LEVEL-0)

R_{JQYJN} Where a I/O Controller Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Realm memory trust boundary (see Trust Boundaries).
(LEVEL-0)

R_{KDXCW} Where a I/O Controller Chiplet is part of a system that is compliant with the Arm Confidential Compute Architecture (CCA) [5], the chiplet is wholly outside the Root memory trust boundary (see Trust Boundaries).
(LEVEL-0)

R_{GRYYV} A I/O Controller Chiplet contains one or more trusted security agents (see Trusted Security Agent) that collectively implement, at a minimum, the following Hardware Enforced Security (HES) functions:

- Secure storage of a chiplet identifier.
- Enforcement of the hardware security lifecycle of the chiplet, including the moderation of invasive subsystems such as debug. The enforcement is the effect of the security lifecycle management that might be undertaken by this Trusted Security Agent or delegated to another Trusted Security Agent.

See Trusted Security Agent for further information on HES.
(LEVEL-0)

R_{JDSWQ} A trusted security agent in a I/O Controller Chiplet is a secondary trusted security agent.
(LEVEL-0)

4.10.1.11 System Counter

I_{KMDKJ} CoreSight trace sources typically include a timestamp in the trace stream. The Arm CoreSight Architecture [7] recommends that designs share the same source of time for both PEs and CoreSight components. Therefore, a I/O Controller Chiplet might contain a system counter that provides the timestamp value.

R_{HWYHJ} Where a I/O Controller Chiplet contains a source of CoreSight trace data, it contains a system counter.
(LEVEL-0)

R_{NVSMJ} A system counter that is in a I/O Controller Chiplet is a secondary system counter.
(LEVEL-0)

4.10.1.12 Debug

I_{NRYBB} A I/O Controller Chiplet contains debug components that are associated with any Application PEs, system controllers, trusted security agents etc.

R_{NHPSD} The debug components contained in a I/O Controller Chiplet can be accessed by a debug access port within the chiplet or in other chiplets in the system.
(LEVEL-0)

R_{PBFBD} A I/O Controller Chiplet supports debug cross triggering. The chiplet supports the transmission of debug trigger events generated by sources in the chiplet to other chiplets in the system, and it also receives debug trigger events generated by sources that are in other chiplets.
(LEVEL-0)

4.10.1.13 Trace

I_{WLJCB} The trace functionality of an Arm system requires that trace data can be routed to and stored in system main memory. CoreSight trace data is routed from sources in a chiplet to a CoreSight *Embedded Trace Router* (ETR) [8] in the same chiplet. The ETR stores the data to memory using normal memory-mapped writes.

R_{GNKGG} Where a I/O Controller Chiplet contains one or more CoreSight trace sources it has a capability to write the trace data collected from components in the chiplet to system main memory. That memory might be in another chiplet in the system.

(LEVEL-0)

4.10.1.14 System

I_{KRZNG} A unique identifier is necessary for a I/O Controller Chiplet. As the system might contain more than one instance of the chiplet, a means of identifying each individual instance in the system is required.

R_{VVBCX} A I/O Controller Chiplet contains an identification register that is accessible to all chiplets in the system. The register holds an identification value that is suitable for use for the determination of the identity of the chiplet instance with regard to the other chiplets in the system and within the scope of the system.

(LEVEL-0)

Chapter 5
Chiplet Interfaces

This chapter describes the interface requirements between different chiplet types.

The interface types are:

- Hub Chiplet to Hub Chiplet Interface.
- Hub Chiplet to Compute Chiplet Interface.
- Compute 1 Chiplet to Compute 1 Chiplet Interface.
- Compute 2 Chiplet to Compute 2 Chiplet Interface.
- Fully Coherent Expansion Chiplet Interface.
- Fully Coherent Expansion (Remote Translation) Chiplet Interface.
- I/O Coherent Expansion (Translated) Chiplet Interface.
- I/O Coherent Expansion (Untranslated) Chiplet Interface.
- I/O Coherent Expansion (Remote Translation) Chiplet Interface.
- I/O Chiplet Interface.
- I/O Controller Chiplet Interface.

5.1 Chiplet Activity States and Interface Availability

I$_{SMSPG}$ Throughout the operational lifecycle of the system the states of chiplet activity and the availability of chiplet interfaces changes. It is useful to the description of interface requirements in this CSA to define certain activity states of chiplets, depicted in Figure 5.1 as an illustrative sequence. These states are later associated with the availability of interfaces in the interface descriptions.

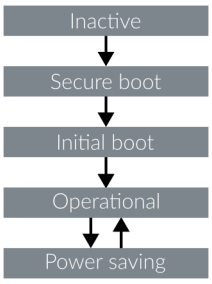

Figure 5.1: Chiplet Activity States

Inactive

The chiplet is inactive. All chiplet interfaces are unavailable.

Secure boot

The security lifecycle states of the trusted security agents are synchronized.

Initial boot

Boot images are loaded to chiplets so that the various security, system control and Application PEs boot or move beyond any initial immutable boot code.

Operational

The system has reached the point in the initialization sequence where the high capacity, relatively high power interfaces between chiplets are available.

Power saving

Some interfaces between chiplets are made unavailable as a power saving measure. Certain communications fall back to relatively low power or dedicated interfaces.

A summary of interface availability by chiplet state is provided in Table 5.1.

Table 5.1: Summary of Chiplet Activity States & Interface Availabilities

Interface	*Inactive*	*Secure boot*	*Initial boot*	*Operational*	*Power saving*
Boot Image Load	Unavailable	Available	Available	Available	Unavailable
Coherent Memory Traffic	Unavailable	Unavailable	Unavailable	Available	Unavailable
Untranslated I/O Coherent Memory Traffic	Unavailable	Unavailable	Unavailable	Available	Unavailable
I/O Coherent Memory Traffic	Unavailable	Unavailable	Unavailable	Available	Unavailable
Non-Coherent Memory Traffic	Unavailable	Unavailable	Unavailable	Available	Unavailable
Device Configuration	Unavailable	Unavailable	Unavailable	Available	Unavailable
Address Translation	Unavailable	Unavailable	Unavailable	Available	Unavailable
Interrupts	Unavailable	Unavailable	Unavailable	Available	Unavailable
GIC Synchronization Messaging	Unavailable	Unavailable	Unavailable	Available	Unavailable
Trusted Security Agent Communication	Unavailable	Available	Available	Available	Available
System Control	Unavailable	Available	Available	Available	Available
System Counter Synchronization	Unavailable	Unavailable	Available	Available	Unavailable
Debug Access	Unavailable	Available	Available	Available	Available
Debug Cross-Trigger	Unavailable	Available	Available	Available	Available
Trace	Unavailable	Unavailable	Unavailable	Available	Unavailable

5.2 Hub Chiplet to Hub Chiplet Interface Requirements

I_{PKTVB} This section specifies the requirements of the interface between a Hub Chiplet and another Hub Chiplet.

An overview of the interfaces between the two chiplets is depicted in Figure 5.2.

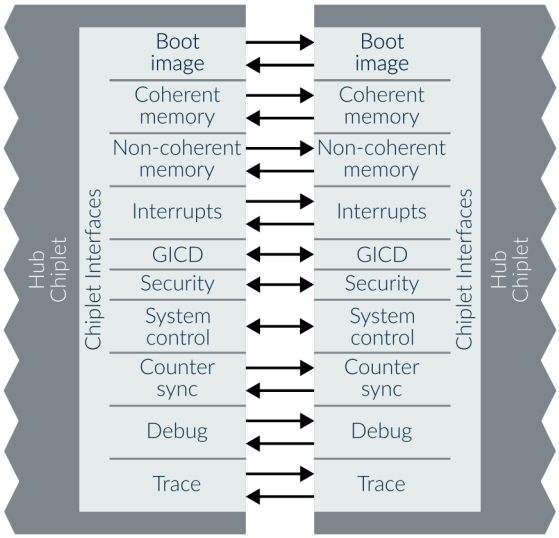

Figure 5.2: Hub Chiplet to Hub Chiplet Interfaces

5.2.1 Boot Image Load Interface

I_{GNTLV} A Boot Image Load Interface between two chiplets that are directly connected transports boot code images from the first chiplet that loads them from non-volatile memory it can access directly or through other chiplets with which is it connected, to the second chiplet and other chiplets connected to that second chiplet.

R_{ZPXZW} The system implementation determines that for a Hub Chiplet and another Hub Chiplet that are directly connected, one chiplet is a source of boot images and the other chiplet is a destination. There is a Boot Image Load Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that is capable of transporting boot images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-0)

R_{SPPXN} A Boot Image Load Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{KMPRP} A Boot Image Load Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{CGLNT} For example, a Boot Image Load Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{SHCBY} For implementation details regarding this interface see Boot Image Load Interface.

5.2.2 Coherent Memory Traffic Interface

I_{XPNJG} A Coherent Memory Traffic Interface allows coherent agents in a Hub Chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in another Hub Chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

R_{SYDRX} There is a Coherent Memory Traffic Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that transports cache coherent memory traffic in both directions.

(LEVEL-0)

R_{LQZMW} A Coherent Memory Traffic Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{JDHFK} For example, a Coherent Memory Traffic Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{VZCKZ} For implementation details regarding this interface see Coherent Memory Traffic Interface.

5.2.3 Non-Coherent Memory Traffic Interface

I_{CBQCQ} A Non-Coherent Memory Traffic Interface allows agents in a Hub Chiplet, and in other chiplets connected to it, to access system addressable memory in another Hub Chiplet that is directly connected, and in other chiplets connected to that other chiplet.

R_{RXQWQ} There is a Non-Coherent Memory Traffic Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that transports non-coherent memory traffic in both directions.

(LEVEL-0)

R_{CCNRY} A Non-Coherent Memory Traffic Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{RHLHH} For example, a Non-Coherent Memory Traffic Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{SQHWQ} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.2.4 Interrupt Handling

I_{ZXQMY} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.2.4.1 Interrupts

R_{DFVLC} There is a Interrupt Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that transports interrupts in both directions.

(LEVEL-0)

R_{CQVLD} A Interrupt Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{JLQMC} For example, a Interrupt Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{ZFVHB} For implementation details regarding this interface see Interrupts.

5.2.4.2 GICD Communication

R_{SSPZM} There is a GICD Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that transports messages for sending data between the GIC Distributors (GICD).

(LEVEL-0)

R_{RSSMW} A GICD Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{HHGPB} For example, a GICD Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{VNRRD} For implementation details regarding this interface see GICD Communication.

5.2.5 Security

R_{DHBHJ} There is a Trusted Security Agent Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{TPSJG} A Trusted Security Agent Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{DKMDM} A Trusted Security Agent Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{RVKPZ} A Trusted Security Agent Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FZSXT} For example, a Trusted Security Agent Communication Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{DVYVQ} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.2.6 System Control Communication

R_{NJPCS} There is a System Control Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{PMLBM} A System Control Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FQPCB} For example, a System Control Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{VFZZK} For implementation details regarding this interface see System Controller Communication.

5.2.7 System Counter Synchronization Interface

R_{RJNMR} There is a System Counter Synchronization Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{VCBTT} A System Counter Synchronization Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{KGCTG} For example, a System Counter Synchronization Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{VVYMN} For implementation details regarding this interface see System Counter Synchronization Interface.

5.2.8 Debug

5.2.8.1 Debug Access

R_{BMRTC} There is a Debug Access Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{BVCMK} For implementation details regarding this interface see Debug Access Interface.

5.2.8.2 Debug Cross-Trigger

R_{YWJQM} There is a Debug Cross-Trigger Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{YYTTD} A Debug Cross-Trigger Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NVRZM} For example, a Debug Cross-Trigger Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{XJYZQ} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.2.9 Trace

R_{RNHMS} There is a Trace Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in either chiplet, or in other chiplets connected to them, to memory locations in either chiplet, or in other chiplets connected to them.

(LEVEL-0)

R_{MRYRH} A Trace Interface between a Hub Chiplet and another Hub Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VZZJQ} For example, a Trace Interface between a Hub Chiplet and another Hub Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{GFQSD} For implementation details regarding this interface see Trace Interface.

5.3 Hub Chiplet to Compute Chiplet Interface Requirements

I_{KPSBM} In this section the name *Compute* is used to refer to the chiplet with which a Hub Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Compute 1 Chiplet or a Compute 2 Chiplet.

I_{WSTSX} An overview of the interfaces between the two chiplets is depicted in Figure 5.3.

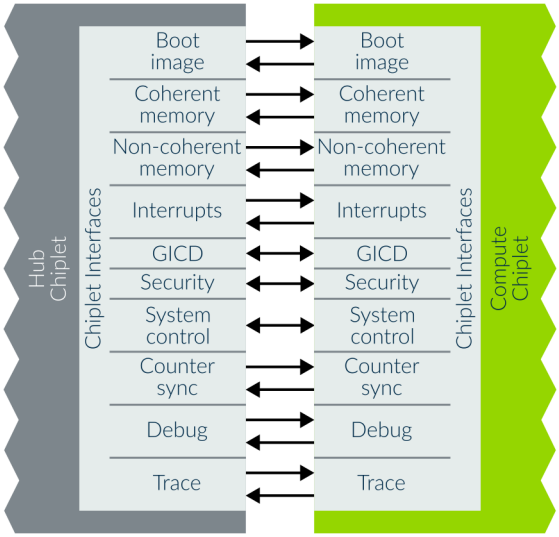

Figure 5.3: Hub Chiplet to Compute Chiplet Interfaces

5.3.1 Boot Image Load Interface

I_{ZSFTB} Non-volatile memory can be accessed by a Hub Chiplet. A Boot Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{NQBSN} The system implementation determines that for a Hub Chiplet and a compute chiplet that are directly connected, one chiplet is a source of boot images and the other chiplet is a destination. There is a Boot Image Load Interface between a Hub Chiplet and a compute chiplet that is directly connected, that is capable of transporting boot images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-0)

R_{SPXCX} A Boot Image Load Interface between a Hub Chiplet and a compute chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{YWSTJ} A Boot Image Load Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{CCDDY} For example, a Boot Image Load Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{WVMBX} For implementation details regarding this interface see Boot Image Load Interface.

5.3.2 Coherent Memory Traffic Interface

I_{GRRLM} A Coherent Memory Traffic Interface allows coherent agents in a Hub Chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in a compute chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

I_{MKFMT} A Coherent Memory Traffic Interface allows coherent agents in a compute chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in a Hub Chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

R_{FVFQQ} There is a Coherent Memory Traffic Interface between a Hub Chiplet and a compute chiplet that is directly connected, that transports cache coherent memory traffic in both directions.

(LEVEL-0)

R_{GHLHT} A Coherent Memory Traffic Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FWCYL} For example, a Coherent Memory Traffic Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{GTCPJ} For implementation details regarding this interface see Coherent Memory Traffic Interface.

5.3.3 Non-Coherent Memory Traffic Interface

I_{FMPMK} A Non-Coherent Memory Traffic Interface allows agents in a Hub Chiplet, and in other chiplets connected to it, to access system addressable memory in a compute chiplet that is directly connected, and in other chiplets connected to that other chiplet.

I_{BWCSM} A Non-Coherent Memory Traffic Interface allows agents in a compute chiplet, and in other chiplets connected to it, to access system addressable memory in a Hub Chiplet that is directly connected, and in other chiplets connected to that other chiplet.

R_{YHFBR} There is a Non-Coherent Memory Traffic Interface between a Hub Chiplet and a compute chiplet that is directly connected, that transports non-coherent memory traffic in both directions.

(LEVEL-0)

R_{YPLLB} A Non-Coherent Memory Traffic Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{ZTJQK} For example, a Non-Coherent Memory Traffic Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{PLVNJ} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.3.4 Interrupt Handling

I_{FGYNK} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.3.4.1 Interrupts

R_{CLWWD} There is a Interrupt Interface between a Hub Chiplet and a compute chiplet that is directly connected, that transports interrupts in both directions.

(LEVEL-0)

R_{GMZTH} A Interrupt Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{YVRLS} For example, a Interrupt Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{PDGKJ} For implementation details regarding this interface see Interrupts.

5.3.4.2 GICD Communication

R_{ZYSFB} There is a GICD Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected, that transports messages for sending data between the GIC Distributors (GICD).

(LEVEL-0)

R_{DQDKT} A GICD Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VGSYR} For example, a GICD Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{YXBLN} For implementation details regarding this interface see GICD Communication.

5.3.5 Security

R_{NNQCD} There is a Trusted Security Agent Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{FXRMW} A Trusted Security Agent Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{SZZGL} A Trusted Security Agent Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{YGWKV} A Trusted Security Agent Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FDWDK} For example, a Trusted Security Agent Communication Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{GWHRY} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.3.6 System Control Communication

R_{BHDVN} There is a System Control Interface between a Hub Chiplet and a compute chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{HJZYS} A System Control Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QBZLG} For example, a System Control Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{FVKJS} For implementation details regarding this interface see System Controller Communication.

5.3.7 System Counter Synchronization Interface

R_{HXDQV} There is a System Counter Synchronization Interface between a Hub Chiplet and a compute chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{JNCKG} A System Counter Synchronization Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{BDMBC} For example, a System Counter Synchronization Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{FYXRR} For implementation details regarding this interface see System Counter Synchronization Interface.

5.3.8 Debug

5.3.8.1 Debug Access

R_{HZCKZ} There is a Debug Access Interface between a Hub Chiplet and a compute chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{BJKDZ} For implementation details regarding this interface see Debug Access Interface.

5.3.8.2 Debug Cross-Trigger

R_{GDTQB} There is a Debug Cross-Trigger Interface between a Hub Chiplet and a compute chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{CJCVS} A Debug Cross-Trigger Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{RRCWN} For example, a Debug Cross-Trigger Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{YDNLZ} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.3.9 Trace

R_{MSRQF} There is a Trace Interface between a Hub Chiplet and a compute chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a compute chiplet, or in other chiplets connected to that chiplet, to memory locations in a Hub Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{SPMCF} Where a compute chiplet contains system main memory, there is a Trace Interface between a compute chiplet and a Hub Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a Hub Chiplet, or in other chiplets connected to that chiplet, to memory locations in a compute chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{WFTMD} A Trace Interface between a Hub Chiplet and a compute chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{HTQFC} For example, a Trace Interface between a Hub Chiplet and a compute chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{MWXPS} For implementation details regarding this interface see Trace Interface.

5.4 Compute 1 Chiplet to Compute 1 Chiplet Interface Requirements

I_{FGFYM} This section specifies the requirements of the interface between a Compute 1 Chiplet and another Compute 1 Chiplet.

An overview of the interfaces between the two chiplets is depicted in Figure 5.4.

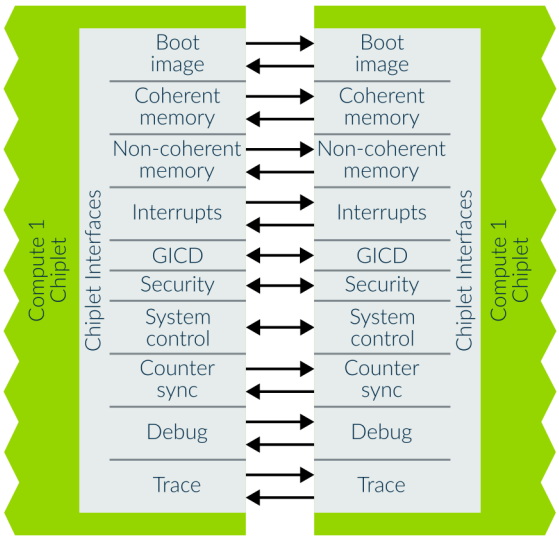

Figure 5.4: Compute 1 Chiplet to Compute 1 Chiplet Interfaces

5.4.1 Boot Image Load Interface

I_{FZTRR} A Boot Image Load Interface between two chiplets that are directly connected transports boot code images from the first chiplet that loads them from non-volatile memory it can access directly or through other chiplets with which is it connected, to the second chiplet and other chiplets connected to that second chiplet.

R_{SZTCH} The system implementation determines that for a Compute 1 Chiplet and another Compute 1 Chiplet that are directly connected, one chiplet is a source of boot images and the other chiplet is a destination. There is a Boot Image Load Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that is capable of transporting boot images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-0)

R_{PMVRY} A Boot Image Load Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{RSGWB} A Boot Image Load Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{LMNCN} For example, a Boot Image Load Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{CZBJL} For implementation details regarding this interface see Boot Image Load Interface.

5.4.2 Coherent Memory Traffic Interface

I_{QHDGN} A Coherent Memory Traffic Interface allows coherent agents in a Compute 1 Chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in another Compute 1 Chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

R_{LPBNG} There is a Coherent Memory Traffic Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that transports cache coherent memory traffic in both directions.

(LEVEL-0)

R_{KYGBK} A Coherent Memory Traffic Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DWPQF} For example, a Coherent Memory Traffic Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{WRXSD} For implementation details regarding this interface see Coherent Memory Traffic Interface.

5.4.3 Non-Coherent Memory Traffic Interface

I_{GJJHN} A Non-Coherent Memory Traffic Interface allows agents in a Compute 1 Chiplet, and in other chiplets connected to it, to access system addressable memory in another Compute 1 Chiplet that is directly connected, and in other chiplets connected to that other chiplet.

R_{RBVXV} There is a Non-Coherent Memory Traffic Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that transports non-coherent memory traffic in both directions.

(LEVEL-0)

R_{TKLSJ} A Non-Coherent Memory Traffic Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FTTZT} For example, a Non-Coherent Memory Traffic Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{CYTWG} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.4.4 Interrupt Handling

I_{PXSQW} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.4.4.1 Interrupts

R_{SDKFD} There is a Interrupt Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that transports interrupts in both directions.

(LEVEL-0)

R_{JSDGJ} A Interrupt Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{WDTFC} For example, a Interrupt Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{DTHZB} For implementation details regarding this interface see Interrupts.

5.4.4.2 GICD Communication

R_{QBRTM} There is a GICD Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that transports messages for sending data between the GIC Distributors (GICD).

(LEVEL-0)

R_{BPTLC} A GICD Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{YXSTZ} For example, a GICD Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{CKQBR} For implementation details regarding this interface see GICD Communication.

5.4.5 Security

R_{KFDDX} There is a Trusted Security Agent Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected. This interface transports messages between the trusted security agents of the chiplets.

(LEVEL-0)

R_{YYBCT} A Trusted Security Agent Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{LPQPB} A Trusted Security Agent Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{SSDSP} A Trusted Security Agent Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QTZGF} For example, a Trusted Security Agent Communication Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{SYDWB} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.4.6 System Control Communication

R_{FZBGN} There is a System Control Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected. This interface transports messages between the system controllers of the chiplets.

(LEVEL-0)

R_{SCNVZ} A System Control Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NMJSN} For example, a System Control Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{LNBDP} For implementation details regarding this interface see System Controller Communication.

5.4.7 System Counter Synchronization Interface

R_{ZRCQC} There is a System Counter Synchronization Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that is capable of synchronizing the system counters of the chiplets.

(LEVEL-0)

R_{DSHTS} A System Counter Synchronization Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{SJYFM} For example, a System Counter Synchronization Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{XHBSH} For implementation details regarding this interface see System Counter Synchronization Interface.

5.4.8 Debug

5.4.8.1 Debug Access

R_{MFVBR} There is a Debug Access Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{GBJSZ} For implementation details regarding this interface see Debug Access Interface.

5.4.8.2 Debug Cross-Trigger

R_{YMRQK} There is a Debug Cross-Trigger Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{TKKSY} A Debug Cross-Trigger Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{GWZFH} For example, a Debug Cross-Trigger Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{MLSCD} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.4.9 Trace

R_{LSFZQ} There is a Trace Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in another Compute 1 Chiplet, or in other chiplets connected to that chiplet, to memory locations in a Compute 1 Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{MSMMY} Where another Compute 1 Chiplet contains system main memory, there is a Trace Interface between another Compute 1 Chiplet and a Compute 1 Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a Compute 1 Chiplet, or in other chiplets connected to that chiplet, to memory locations in another Compute 1 Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{PHLYM} A Trace Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DHQHZ} For example, a Trace Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{NTPFR} For implementation details regarding this interface see Trace Interface.

5.5 Compute 2 Chiplet to Compute 2 Chiplet Interface Requirements

I_{HRJYK} This section specifies the requirements of the interface between a Compute 2 Chiplet and another Compute 2 Chiplet.

An overview of the interfaces between the two chiplets is depicted in Figure 5.5.

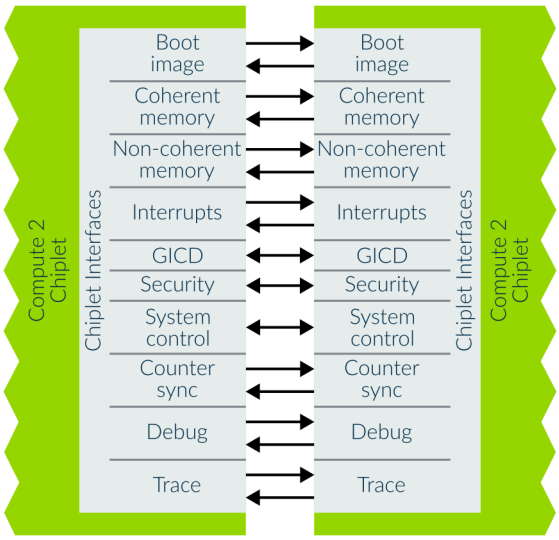

Figure 5.5: Compute 2 Chiplet to Compute 2 Chiplet Interfaces

5.5.1 Boot Image Load Interface

I_{SBNTD} A Boot Image Load Interface between two chiplets that are directly connected transports boot code images from the first chiplet that loads them from non-volatile memory it can access directly or through other chiplets with which is it connected, to the second chiplet and other chiplets connected to that second chiplet.

R_{JRSXS} The system implementation determines that for a Compute 2 Chiplet and another Compute 2 Chiplet that are directly connected, one chiplet is a source of boot images and the other chiplet is a destination. There is a Boot Image Load Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that is capable of transporting boot images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-0)

R_{MKCPL} A Boot Image Load Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{RLQYC} A Boot Image Load Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VSRWY} For example, a Boot Image Load Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{XFJML} For implementation details regarding this interface see Boot Image Load Interface.

5.5.2 Coherent Memory Traffic Interface

I_{KBDPD} A Coherent Memory Traffic Interface allows coherent agents in a Compute 2 Chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in another Compute 2 Chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

R_{RBNSS} There is a Coherent Memory Traffic Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that transports cache coherent memory traffic in both directions.

(LEVEL-0)

R_{CZLQD} A Coherent Memory Traffic Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{KGYZK} For example, a Coherent Memory Traffic Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{MNYKS} For implementation details regarding this interface see Coherent Memory Traffic Interface.

5.5.3 Non-Coherent Memory Traffic Interface

I_{YDXZL} A Non-Coherent Memory Traffic Interface allows agents in a Compute 2 Chiplet, and in other chiplets connected to it, to access system addressable memory in another Compute 2 Chiplet that is directly connected, and in other chiplets connected to that other chiplet.

R_{SKNMG} There is a Non-Coherent Memory Traffic Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that transports non-coherent memory traffic in both directions.

(LEVEL-0)

R_{BKKWV} A Non-Coherent Memory Traffic Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DFDQS} For example, a Non-Coherent Memory Traffic Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{RXFXZ} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.5.4 Interrupt Handling

I_{TWRML} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.5.4.1 Interrupts

R_{NLNCD} There is a Interrupt Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that transports interrupts in both directions.

(LEVEL-0)

R_{JRLRN} A Interrupt Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{ZLJZD} For example, a Interrupt Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{ZWVHK} For implementation details regarding this interface see Interrupts.

5.5.4.2 GICD Communication

R_{DSNRH} There is a GICD Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that transports messages for sending data between the GIC Distributors (GICD).

(LEVEL-0)

R_{SBJDB} A GICD Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{JQHJZ} For example, a GICD Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{KTTGS} For implementation details regarding this interface see GICD Communication.

5.5.5 Security

R_{TXHMX} There is a Trusted Security Agent Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{LLTTW} A Trusted Security Agent Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{GZMKP} A Trusted Security Agent Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{RTCSK} A Trusted Security Agent Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NCQLX} For example, a Trusted Security Agent Communication Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{PRZWQ} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.5.6 System Control Communication

R_{PDJZG} There is a System Control Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{NLZFW} A System Control Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{GVNWK} For example, a System Control Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{GFBQC} For implementation details regarding this interface see System Controller Communication.

5.5.7 System Counter Synchronization Interface

R_{JRCFK} There is a System Counter Synchronization Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{WYDPS} A System Counter Synchronization Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{LJSWG} For example, a System Counter Synchronization Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{PFKBL} For implementation details regarding this interface see System Counter Synchronization Interface.

5.5.8 Debug

5.5.8.1 Debug Access

R_{QLHKH} There is a Debug Access Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{YGPRL} For implementation details regarding this interface see Debug Access Interface.

5.5.8.2 Debug Cross-Trigger

R_{QVHXK} There is a Debug Cross-Trigger Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{HNFBT} A Debug Cross-Trigger Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{ZXKYY} For example, a Debug Cross-Trigger Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{HRVZS} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.5.9 Trace

R_{QTVRH} There is a Trace Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in either chiplet, or in other chiplets connected to them, to memory locations in either chiplet, or in other chiplets connected to them.

(LEVEL-0)

R_{RXSYB} A Trace Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{SWNVJ} For example, a Trace Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{PBMPM} For implementation details regarding this interface see Trace Interface.

5.6 Fully Coherent Expansion Chiplet Interface Requirements

I_{DJFXG} In this section the name *Host* is used to refer to the chiplet with which a Fully Coherent Expansion Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, or another Fully Coherent Expansion Chiplet.

I_{FYFCP} An overview of the interfaces between the two chiplets is depicted in Figure 5.6.

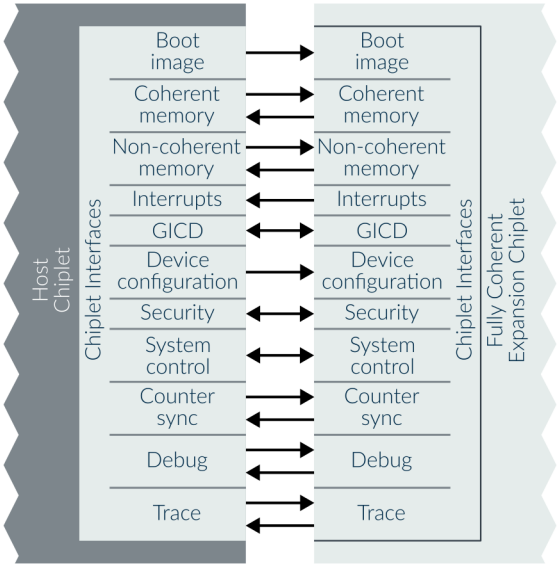

Figure 5.6: Fully Coherent Expansion Chiplet Interfaces

5.6.1 Boot Image Load Interface

I_{YDTNL} Non-volatile memory can be accessed by a host chiplet. A Boot Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{PRKPY} There is a Boot Image Load Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that is capable of transporting boot images from a host chiplet to memory locations in a Fully Coherent Expansion Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{VFPKB} A Boot Image Load Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{WPZNH} A Boot Image Load Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{LQCLJ} For example, a Boot Image Load Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{RDPDB} For implementation details regarding this interface see Boot Image Load Interface.

5.6.2 Coherent Memory Traffic Interface

I_{GRSRV} A Coherent Memory Traffic Interface allows coherent agents in a host chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in a Fully Coherent Expansion Chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

I_{BTYCX} A Coherent Memory Traffic Interface allows coherent agents in a Fully Coherent Expansion Chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in a host chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

R_{VLQPR} There is a Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that transports cache coherent memory traffic in both directions.

(LEVEL-0)

R_{RBMCH} A Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{ZYKQY} For example, a Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{BHBCH} For implementation details regarding this interface see Coherent Memory Traffic Interface.

5.6.3 Non-Coherent Memory Traffic Interface

I_{BJSNC} A Non-Coherent Memory Traffic Interface allows agents in a host chiplet to access system addressable memory in a Fully Coherent Expansion Chiplet that is directly connected, and in chiplets connected to that chiplet.

I_{HRVGG} A Non-Coherent Memory Traffic Interface allows agents in a Fully Coherent Expansion Chiplet to access system addressable memory in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{NTBDH} There is a Non-Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that transports non-coherent memory traffic in both directions.

(LEVEL-0)

R_{QTGWK} A Non-Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FBJSB} For example, a Non-Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{SXWVM} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.6.4 Interrupt Handling

I_{MDKRR} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.6.4.1 Interrupts

R_{WGSPV} There is a Interrupt Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that transports interrupts in both directions.

(LEVEL-0)

R_{PFZGF} A Interrupt Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{XZQWV} For example, a Interrupt Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{QGBCN} For implementation details regarding this interface see Interrupts.

5.6.4.2 GICD Communication

R_{XVQPJ} There is a GICD Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that transports messages for sending data between the GIC Distributors (GICD).

(LEVEL-0)

R_{BYBMM} A GICD Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QWMBC} For example, a GICD Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{GZXSM} For implementation details regarding this interface see GICD Communication.

5.6.5 Device Configuration Interface

R_{GJBYM} There is a Device Configuration Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that allows host software to configure the device(s) in a Fully Coherent Expansion Chiplet.

(LEVEL-0)

R_{JZNWW} A Device Configuration Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{CGBPH} For example, a Device Configuration Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{RVVGW} For implementation details regarding this interface see Device Configuration Interface.

5.6.6 Security

R_{JZKYZ} There is a Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{RVDHX} A Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{ZCQQH} A Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{GHGXC} A Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{MBYQJ} For example, a Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{VYLSW} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.6.7 System Control Communication

R_{GQTVW} There is a System Control Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{LJRBQ} A System Control Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QMLZC} For example, a System Control Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{HQGGR} For implementation details regarding this interface see System Controller Communication.

5.6.8 System Counter Synchronization

R_{YMVWW} Where a Fully Coherent Expansion Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{KMLZL} A System Counter Synchronization Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{CTNVV} For example, a System Counter Synchronization Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{HGQXD} For implementation details regarding this interface see System Counter Synchronization Interface.

5.6.9 Debug

5.6.9.1 Debug Access

R_{FZVDD} There is a Debug Access Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{RDPLD} For implementation details regarding this interface see Debug Access Interface.

5.6.9.2 Debug Cross-Trigger

R_{RZHPT} There is a Debug Cross-Trigger Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{YDDYT} A Debug Cross-Trigger Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DKGNG} For example, a Debug Cross-Trigger Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{GBSMX} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.6.10 Trace

R_{BNFXW} Where a Fully Coherent Expansion Chiplet contains a source of CoreSight trace data, or it facilitates the transfer of trace data to memory from a chiplet that it is connected to, there is a Trace Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a Fully Coherent Expansion Chiplet, or in other chiplets connected to that chiplet, to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{JSNRX} Where a Fully Coherent Expansion Chiplet contains system main memory, there is a Trace Interface between a Fully Coherent Expansion Chiplet and a host chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a host chiplet, or in other chiplets connected to that chiplet, to memory locations in a Fully Coherent Expansion Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{HYMKW} A Trace Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{GCZBX} For example, a Trace Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{GDLQM} For implementation details regarding this interface see Trace Interface.

5.7 Fully Coherent Expansion (Remote Translation) Chiplet Interface Requirements

I_{HHFHD} In this section the name *Host* is used to refer to the chiplet to which a Fully Coherent Expansion (Remote Translation) Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, or a Fully Coherent Expansion Chiplet.

I_{BHKFR} An overview of the interfaces between the two chiplets is depicted in Figure 5.7.

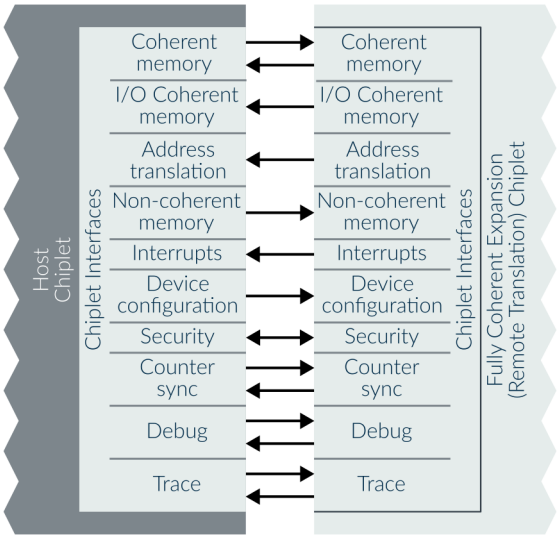

Figure 5.7: Fully Coherent Expansion (Remote Translation) Chiplet Interfaces

5.7.1 Coherent Memory Traffic Interface

I_{GPSBW} Coherent Memory Traffic Interface allows coherent agents in a host chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

I_{MNQGD} Coherent Memory Traffic Interface allows coherent agents in a Fully Coherent Expansion (Remote Translation) Chiplet, and in other chiplets connected to it, access to system addressable memory and coherent caches in a host chiplet that is directly connected, and in other chiplets connected to that other chiplet. In addition to these accesses the relevant snoop, DVM, exclusives and MPAM messages are also transported.

R_{CMRXD} There is a Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports cache coherent memory traffic in both directions.
(LEVEL-0)

R_{TGCLP} A Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.
(LEVEL-0)

U_{GXKGF} For example, a Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{MFGBW} For implementation details regarding this interface see Fully Coherent Memory Traffic Interface.

5.7.2 I/O Coherent Memory Traffic Interface

I_{ZKZJZ} A I/O Coherent Memory Traffic Interface allows I/O coherent agents in a Fully Coherent Expansion (Remote Translation) Chiplet to access system addressable memory and coherent caches in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{QSHDG} There is a I/O Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports I/O coherent memory traffic from a Fully Coherent Expansion (Remote Translation) Chiplet to a host chiplet.

(LEVEL-0)

R_{SXWZM} A I/O Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QDKVT} For example, a I/O Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{GTCRB} For implementation details regarding this interface see I/O Coherent Memory Traffic Interface.

5.7.3 Address Translation Interface

R_{VBVKL} There is a Address Translation Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that allows device(s) in a Fully Coherent Expansion (Remote Translation) Chiplet to access address translations through an MMU (such as a Arm SMMU [9]) in a host chiplet.

(LEVEL-0)

R_{LHSRL} A Address Translation Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NWGWD} For example, a Address Translation Interfacebetween a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{YBZRH} For implementation details regarding this interface see Address Translation Interface.

5.7.4 Non-Coherent Memory Traffic Interface

I_{FVKYQ} A Non-Coherent Memory Traffic Interface allows agents in a host chiplet to access system addressable memory in a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{NKXXM} There is a Non-Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports non-coherent memory traffic from a host chiplet to a Fully Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

R_{NBWLZ} A Non-Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QNVPS} For example, a Non-Coherent Memory Traffic Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{MHJQY} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.7.5 Interrupt Handling

I_{ZHFTY} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.7.5.1 Interrupts

I_{BJLXZ} A Fully Coherent Expansion (Remote Translation) Chiplet might contain components that generate interrupts.

R_{LWJBR} There is a Interrupt Interface from a host chiplet to a Fully Coherent Expansion (Remote Translation) Chiplet that transports interrupts, as MSIs, from a Fully Coherent Expansion (Remote Translation) Chiplet to a GIC ITS [4] in a host chiplet.

(LEVEL-0)

R_{HYTDS} A Interrupt Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{BDRFG} For example, a Interrupt Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{CYSDK} For implementation details regarding this interface see Interrupts.

5.7.6 Device Configuration Interface

R_{GLVZD} There is a Device Configuration Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that allows host software to configure the device(s) in a Fully Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

R_{LXTVM} A Device Configuration Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{BYXHS} For example, a Device Configuration Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{YYTZY} For implementation details regarding this interface see Device Configuration Interface.

5.7.7 Security

R_{GRXQH} There is a Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{CSFZB} A Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{GGTMJ} A Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{QBXFV} A Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{WGCCL} For example, a Trusted Security Agent Communication Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{TYFDG} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.7.8 System Counter Synchronization

R_{BMWVT} Where a Fully Coherent Expansion (Remote Translation) Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{FWJVY} A System Counter Synchronization Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NJTZS} For example, a System Counter Synchronization Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{NRGFF} For implementation details regarding this interface see System Counter Synchronization Interface.

5.7.9 Debug

5.7.9.1 Debug Access

R_{PCWPV} There is a Debug Access Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{WKZWQ} For implementation details regarding this interface see Debug Access Interface.

5.7.9.2 Debug Cross-Trigger

R_{WWKGM} There is a Debug Cross-Trigger Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{FVSKN} A Debug Cross-Trigger Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{LZCWG} For example, a Debug Cross-Trigger Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{KCPFS} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.7.10 Trace

R_{YBDFB} Where a Fully Coherent Expansion (Remote Translation) Chiplet contains a CoreSight trace source, there is a Trace Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a Fully Coherent Expansion (Remote Translation) Chiplet to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{VSTHC} Where a Fully Coherent Expansion (Remote Translation) Chiplet contains system main memory, there is a Trace Interface between a Fully Coherent Expansion (Remote Translation) Chiplet and a host chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a host chiplet, or in other chiplets connected to that chiplet, to memory locations in a Fully Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

R_{WKLYR} A Trace Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{ZHBXD} For example, a Trace Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{NFMYP} For implementation details regarding this interface see Trace Interface.

5.8 I/O Coherent Expansion (Translated) Chiplet Interface Requirements

I_{QJSSD} In this section the name *Host* is used to refer to the chiplet with which a I/O Coherent Expansion (Translated) Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, a Fully Coherent Expansion Chiplet or another I/O Coherent Expansion (Translated) Chiplet.

I_{JZCPB} An overview of the interfaces between the two chiplets is depicted in Figure 5.8.

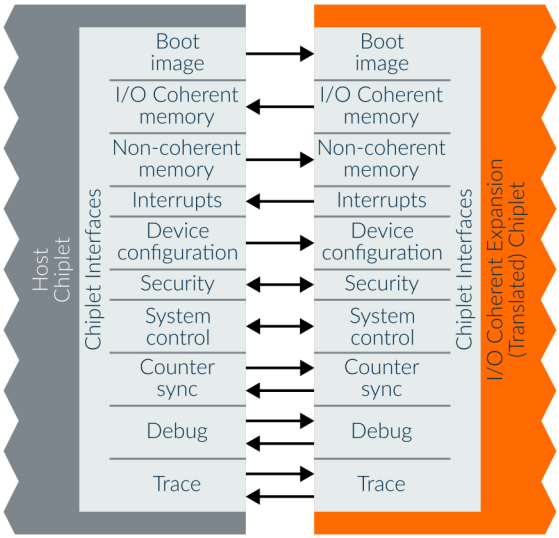

Figure 5.8: I/O Coherent Expansion (Translated) Chiplet Interfaces

5.8.1 Boot Image Load Interface

I_{SLXGZ} Non-volatile memory can be accessed by a host chiplet. A Boot Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{CNCGH} There is a Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that is capable of transporting boot images from a host chiplet to memory locations in a I/O Coherent Expansion (Translated) Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{XTJPP} A Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{JLGGN} A Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VRZYT} For example, a Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{FZGMS} For implementation details regarding this interface see Boot Image Load Interface.

5.8.2 I/O Coherent Memory Traffic Interface

I_{LCPND} A I/O Coherent Memory Traffic Interface allows I/O coherent agents in a I/O Coherent Expansion (Translated) Chiplet to access system addressable memory and coherent caches in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{HCYKF} There is a I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that transports I/O coherent memory traffic from a I/O Coherent Expansion (Translated) Chiplet to a host chiplet.

(LEVEL-0)

R_{HXNJH} A I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{XVPJB} For example, a I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{GWWSK} For implementation details regarding this interface see I/O Coherent Memory Traffic Interface.

5.8.3 Non-Coherent Memory Traffic Interface

I_{BFSDB} A Non-Coherent Memory Traffic Interface allows agents in a host chiplet to access system addressable memory in a I/O Coherent Expansion (Translated) Chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{RGQKJ} There is a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that transports non-coherent memory traffic from a host chiplet to a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

R_{YHSGY} A Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{STBSZ} For example, a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{KRTRC} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.8.4 Interrupt Handling

I_{JCQGS} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.8.4.1 Interrupts

I_{YZDGH} A I/O Coherent Expansion (Translated) Chiplet might contain components that generate interrupts.

R_{JSFTT} There is a Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that transports interrupts, as MSI-64, from a I/O Coherent Expansion (Translated) Chiplet to a GIC ITS [4] in a host chiplet.

(LEVEL-0)

R_{BMJZP} A Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DCVYN} For example, a Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{WVVWQ} For implementation details regarding this interface see Interrupts.

5.8.5 Device Configuration Interface

R_{RNYKY} There is a Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that allows host software to configure the device(s) that are in a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

I_{GKSGZ} For implementation details regarding this interface see Device Configuration Interface.

5.8.6 Security

R_{XRDCX} There is a Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{ZGDGY} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{JJKSQ} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{BHRHJ} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{JXQZC} For example, a Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{GXDBS} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.8.7 System Control Communication

R_{NTZHB} There is a System Control Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{SQLJT} A System Control Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{BGPYB} For example, a System Control Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{ZRKPF} For implementation details regarding this interface see System Controller Communication.

5.8.8 System Counter Synchronization

R_{CMRMD} Where a I/O Coherent Expansion (Translated) Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{WDLDK} A System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{CYHGF} For example, a System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{RMZZS} For implementation details regarding this interface see System Counter Synchronization Interface.

5.8.9 Debug

5.8.9.1 Debug Access

R_{CYHBV} There is a Debug Access Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{WZWPM} For implementation details regarding this interface see Debug Access Interface.

5.8.9.2 Debug Cross-Trigger

R_{GRCDW} There is a Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{ZNNHF} A Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{HDNGH} For example, a Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{PZZKG} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.8.10 Trace

R_{NTBDN} Where a I/O Coherent Expansion (Translated) Chiplet contains a source of CoreSight trace data, or it facilitates the transfer of trace data to memory from a chiplet that it is connected to, there is a Trace Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a I/O Coherent Expansion (Translated) Chiplet, or in other chiplets connected to that chiplet, to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{RMTPR} Where a I/O Coherent Expansion (Translated) Chiplet contains system main memory, there is a Trace Interface between a I/O Coherent Expansion (Translated) Chiplet and a host chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a host chiplet, or in other chiplets connected to that chiplet, to memory locations in a I/O Coherent Expansion (Translated) Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{ZCDGQ} A Trace Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{MSQPG} For example, a Trace Interface between a host chiplet and a I/O Coherent Expansion (Translated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{CPKLV} For implementation details regarding this interface see Trace Interface.

5.9 I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements

IGZLLQ In this section the name *Host* is used to refer to the chiplet with which a I/O Coherent Expansion (Untranslated) Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, a Fully Coherent Expansion Chiplet, a I/O Coherent Expansion (Translated) Chiplet or another I/O Coherent Expansion (Untranslated) Chiplet.

IQGNZM An overview of the interfaces between the two chiplets is depicted in Figure 5.9.

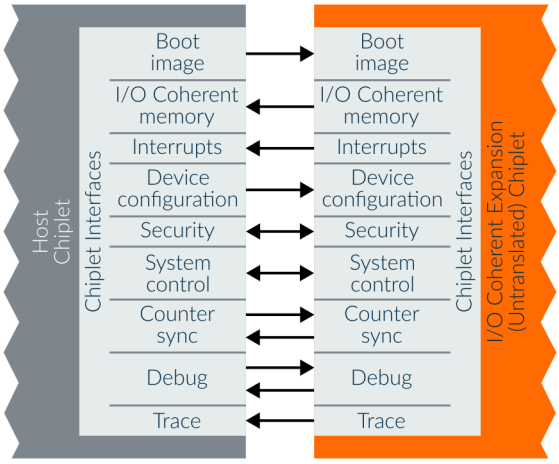

Figure 5.9: I/O Coherent Expansion (Untranslated) Chiplet Interfaces

5.9.1 Boot Image Load Interface

IMHJVS Non-volatile memory can be accessed by a host chiplet. A Boot Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

RWBSBR There is a Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that is capable of transporting boot images from a host chiplet to memory locations in a I/O Coherent Expansion (Untranslated) Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

RTJPJV A Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

RWWJWH A Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

UNMRMC For example, a Boot Image Load Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

IDXGYQ For implementation details regarding this interface see Boot Image Load Interface.

5.9.2 Untranslated I/O Coherent Memory Traffic Interface

I_{CPDTB} A Untranslated I/O Coherent Memory Traffic Interface allows I/O coherent agents in a I/O Coherent Expansion (Untranslated) Chiplet to access system addressable memory and coherent caches in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{CBWZD} There is a Untranslated I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that transports I/O coherent memory traffic from a I/O Coherent Expansion (Untranslated) Chiplet to a host chiplet.

(LEVEL-0)

R_{SZXHJ} A Untranslated I/O Coherent Memory Traffic Interface from a I/O Coherent Expansion (Untranslated) Chiplet to a host chiplet that is directly connected, is untranslated.

(LEVEL-0)

R_{QBQMD} A Untranslated I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{WCRTK} For example, a Untranslated I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{TPZJH} For implementation details regarding this interface see I/O Coherent Memory Traffic Interface.

5.9.3 Interrupt Handling

I_{SDGFD} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.9.3.1 Interrupts

I_{PDLGP} A I/O Coherent Expansion (Untranslated) Chiplet might contain components that generate interrupts.

R_{DTDLR} There is a Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that transports interrupts, as MSIs, from a I/O Coherent Expansion (Untranslated) Chiplet through an MMU to a GIC ITS [4], both in a host chiplet.

(LEVEL-0)

R_{CDBPS} A Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{MVZLP} For example, a Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{BWHKL} For implementation details regarding this interface see Interrupts.

5.9.4 Device Configuration Interface

R_{RYVZX} There is a Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that allows host software to configure the device(s) in a I/O Coherent Expansion (Untranslated) Chiplet.

(LEVEL-0)

R_{QTJWG} A Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FKBXQ} For example, a Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{SHJVQ} For implementation details regarding this interface see Device Configuration Interface.

5.9.5 Security

R_{BSQPC} There is a Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{MCCDT} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{PVDQS} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{PDJPV} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VYRSP} For example, a Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{ZDMFP} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.9.6 System Control Communication

R_{CDVBW} There is a System Control Interface for the system controller in a host chiplet to communicate with the system control agent in a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected.

(LEVEL-0)

R_{ZGNCM} A System Control Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{WMTXJ} For example, a System Control Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{KFTLS} For implementation details regarding this interface see System Controller Communication.

5.9.7 System Counter Synchronization

R_{PPGVB} Where a I/O Coherent Expansion (Untranslated) Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{DJZLN} A System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{TPXJD} For example, a System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{HHPFZ} For implementation details regarding this interface see System Counter Synchronization Interface.

5.9.8 Debug

5.9.8.1 Debug Access

R_{JJPDD} There is a Debug Access Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{VYWHN} For implementation details regarding this interface see Debug Access Interface.

5.9.8.2 Debug Cross-Trigger

R_{JSFJW} There is a Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{JWPNB} A Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{KDPSX} For example, a Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{SQWSJ} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.9.9 Trace

R_{KSYQB} Where a I/O Coherent Expansion (Untranslated) Chiplet contains a CoreSight trace source, there is a Trace Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a I/O Coherent Expansion (Untranslated) Chiplet to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{QXLWL} A Trace Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{LPRWK} For example, a Trace Interface between a host chiplet and a I/O Coherent Expansion (Untranslated) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{SHYXT} For implementation details regarding this interface see Trace Interface.

5.10 I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements

I_{JLVTG} In this section the name *Host* is used to refer to the chiplet to which a I/O Coherent Expansion (Remote Translation) Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, a Fully Coherent Expansion Chiplet or a I/O Coherent Expansion (Translated) Chiplet.

I_{WLPLN} An overview of the interfaces between the two chiplets is depicted in Figure 5.10.

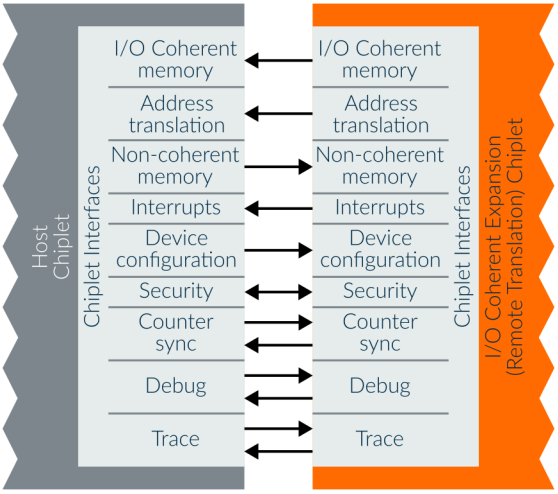

Figure 5.10: I/O Coherent Expansion (Remote Translation) Chiplet Interfaces

5.10.1 I/O Coherent Memory Traffic Interface

I_{HHGDJ} A I/O Coherent Memory Traffic Interface allows I/O coherent agents in a I/O Coherent Expansion (Remote Translation) Chiplet to access system addressable memory and coherent caches in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{HHPXJ} There is a I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports I/O coherent memory traffic from a I/O Coherent Expansion (Remote Translation) Chiplet to a host chiplet.

(LEVEL-0)

R_{YQSDV} A I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{BHWXB} For example, a I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{XQBRP} For implementation details regarding this interface see I/O Coherent Memory Traffic Interface.

5.10.2 Address Translation Interface

R_{MTLNR} There is a Address Translation Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that allows device(s) in a I/O Coherent Expansion (Remote Translation) Chiplet to access address translations through an MMU (such as a Arm SMMU [9]) in a host chiplet.

(LEVEL-0)

R_{SVYMX} A Address Translation Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{MMWGY} For example, a Address Translation Interfacebetween a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{RRBJG} For implementation details regarding this interface see Address Translation Interface.

5.10.3 Non-Coherent Memory Traffic Interface

I_{GVFGW} A Non-Coherent Memory Traffic Interface allows agents in a host chiplet to access system addressable memory in a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{NMQCH} There is a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports non-coherent memory traffic from a host chiplet to a I/O Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

R_{ZLKGN} A Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{GTXWW} For example, a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{PRZTK} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.10.4 Interrupt Handling

I_{HHCLY} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.10.4.1 Interrupts

I_{HMLRC} A I/O Coherent Expansion (Remote Translation) Chiplet might contain components that generate interrupts.

R_{QRJPN} There is a Interrupt Interface from a host chiplet to a I/O Coherent Expansion (Remote Translation) Chiplet that transports interrupts, as MSIs, from a I/O Coherent Expansion (Remote Translation) Chiplet to a GIC ITS [4] in a host chiplet.

(LEVEL-0)

R_{RVRZS} A Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{YXMXV} For example, a Interrupt Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{KTFNX} For implementation details regarding this interface see Interrupts.

5.10.5 Device Configuration Interface

R_{LFDBC} There is a Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that allows host software to configure the device(s) in a I/O Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

R_{ZTFJZ} A Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VCDYR} For example, a Device Configuration Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{WSFJW} For implementation details regarding this interface see Device Configuration Interface.

5.10.6 Security

R_{BNLPG} There is a Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{LGJSW} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{VQBCT} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{FQQRQ} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{LGDTC} For example, a Trusted Security Agent Communication Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{PHDNM} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.10.7 System Counter Synchronization

R_{KDYYR} Where a I/O Coherent Expansion (Remote Translation) Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{MGPHF} A System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{KSCSQ} For example, a System Counter Synchronization Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{XJRFF} For implementation details regarding this interface see System Counter Synchronization Interface.

5.10.8 Debug

5.10.8.1 Debug Access

R_{SDPSD} There is a Debug Access Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{BSCMY} For implementation details regarding this interface see Debug Access Interface.

5.10.8.2 Debug Cross-Trigger

R_{KVYCY} There is a Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{BKNRG} A Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{MRQZQ} For example, a Debug Cross-Trigger Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{GTFCT} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.10.9 Trace

R_{HQWXD} Where a I/O Coherent Expansion (Remote Translation) Chiplet contains a CoreSight trace source, there is a Trace Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a I/O Coherent Expansion (Remote Translation) Chiplet to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{NVKKZ} Where a I/O Coherent Expansion (Remote Translation) Chiplet contains system main memory, there is a Trace Interface between a I/O Coherent Expansion (Remote Translation) Chiplet and a host chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a host chiplet, or in other chiplets connected to that chiplet, to memory locations in a I/O Coherent Expansion (Remote Translation) Chiplet.

(LEVEL-0)

R_{KZNLW} A Trace Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{CBMLY} For example, a Trace Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{WCCHR} For implementation details regarding this interface see Trace Interface.

5.11 I/O Chiplet Interface Requirements

I_{XSVHF} In this section the name *Host* is used to refer to the chiplet with which a I/O Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, a Fully Coherent Expansion Chiplet, a I/O Coherent Expansion (Translated) Chiplet, a I/O Coherent Expansion (Untranslated) Chiplet, a I/O Coherent Expansion (Translated) Chiplet or another I/O Chiplet.

I_{WDCXN} An overview of the interfaces between the two chiplets is depicted in Figure 5.11.

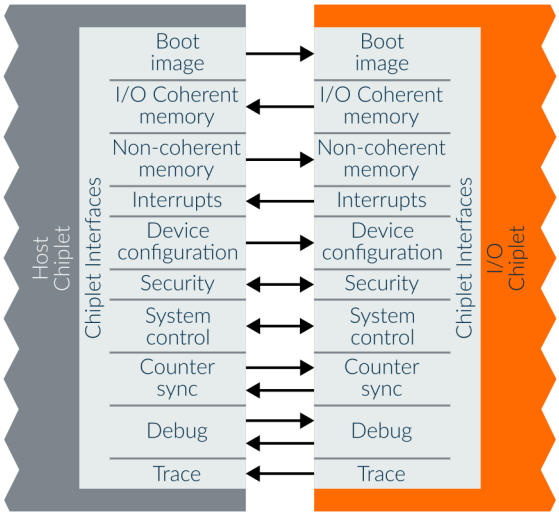

Figure 5.11: I/O Chiplet Interfaces

5.11.1 Boot Image Load Interface

I_{HFCCM} Non-volatile memory can be accessed by a host chiplet. A Boot Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{WGMMT} There is a Boot Image Load Interface between a host chiplet and a I/O Chiplet that is directly connected, that is capable of transporting boot images from a host chiplet to memory locations in a I/O Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{PVTQN} A Boot Image Load Interface between a host chiplet and a I/O Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{GBYYK} A Boot Image Load Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QPMHQ} For example, a Boot Image Load Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{FGHMK} For implementation details regarding this interface see Boot Image Load Interface.

5.11.2 Untranslated I/O Coherent Memory Traffic Interface

I_{QBXCG} A Untranslated I/O Coherent Memory Traffic Interface allows I/O coherent agents in a I/O Chiplet to access system addressable memory and coherent caches in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{LHRDK} There is a Untranslated I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Chiplet that is directly connected, that transports I/O coherent memory traffic from a I/O Chiplet to a host chiplet.

(LEVEL-0)

R_{WLLKW} A Untranslated I/O Coherent Memory Traffic Interface from a I/O Chiplet to a host chiplet that is directly connected, is untranslated.

(LEVEL-0)

R_{XKHXM} A Untranslated I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FYVBW} For example, a Untranslated I/O Coherent Memory Traffic Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{CFKLR} For implementation details regarding this interface see I/O Coherent Memory Traffic Interface.

5.11.3 Non-Coherent Memory Traffic Interface

I_{PCVDS} A Non-Coherent Memory Traffic Interface allows agents in a host chiplet to access system addressable memory in a I/O Chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{DVBZV} There is a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Chiplet that is directly connected, that transports non-coherent memory traffic from a host chiplet to a I/O Chiplet.

(LEVEL-0)

R_{RSJDZ} A Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NRNFZ} For example, a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{STLWC} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.11.4 Interrupt Handling

I_{PBRMC} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.11.4.1 Interrupts

I_{XYFWM} A I/O Chiplet might contain components that generate interrupts.

R_{DFQYV} There is a Interrupt Interface between a host chiplet and a I/O Chiplet that transports interrupts, as MSIs, from a I/O Chiplet through an MMU to a GIC ITS [4], both in a host chiplet.

(LEVEL-0)

R_{PMXFG} A Interrupt Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FRMKJ} For example, a Interrupt Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{PWGNR} For implementation details regarding this interface see Interrupts.

5.11.5 Device Configuration Interface

R_{YLRCM} There is a Device Configuration Interface between a host chiplet and a I/O Chiplet that is directly connected, that allows host software to configure the device(s) in a I/O Chiplet.

(LEVEL-0)

R_{HJDFT} A Device Configuration Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NVBYX} For example, a Device Configuration Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{DKNMD} For implementation details regarding this interface see Device Configuration Interface.

5.11.6 Security

R_{RVTJM} There is a Trusted Security Agent Communication Interface between a host chiplet and a I/O Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{KBGTC} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{HXWGL} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{HHJRZ} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{RVPVJ} For example, a Trusted Security Agent Communication Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{DPMFL} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.11.7 System Control Communication

R_{DDSPX} There is a System Control Interface for the system controller in a host chiplet to communicate with the system control agent in a I/O Chiplet that is directly connected.

(LEVEL-0)

R_{PTQBD} A System Control Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VGPTF} For example, a System Control Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{VGQXP} For implementation details regarding this interface see System Controller Communication.

5.11.8 System Counter Synchronization

R_{LVRMZ} Where a I/O Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a I/O Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{JVJDH} A System Counter Synchronization Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{BTCMH} For example, a System Counter Synchronization Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{YJJKZ} For implementation details regarding this interface see System Counter Synchronization Interface.

5.11.9 Debug

5.11.9.1 Debug Access

R_{TDQTB} There is a Debug Access Interface between a host chiplet and a I/O Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{GCSVJ} For implementation details regarding this interface see Debug Access Interface.

5.11.9.2 Debug Cross-Trigger

R_{XLNKX} There is a Debug Cross-Trigger Interface between a host chiplet and a I/O Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{QHJWB} A Debug Cross-Trigger Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{ZJQVG} For example, a Debug Cross-Trigger Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{SVPNF} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.11.10 Trace

R_{QLRQL} Where a I/O Chiplet contains a CoreSight trace source, there is a Trace Interface between a host chiplet and a I/O Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a I/O Chiplet to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{SRTTF} A Trace Interface between a host chiplet and a I/O Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{YGYDR} For example, a Trace Interface between a host chiplet and a I/O Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{YCRLD} For implementation details regarding this interface see Trace Interface.

5.12 I/O Controller Chiplet Interface Requirements

I_{TCHHQ} In this section the name *Host* is used to refer to the chiplet with which a I/O Controller Chiplet is connected, and the interface with which the requirements in this section apply. That chiplet is a Hub Chiplet, a Compute 2 Chiplet, a Fully Coherent Expansion Chiplet, a I/O Coherent Expansion (Translated) Chiplet, a I/O Coherent Expansion (Untranslated) Chiplet, a I/O Coherent Expansion (Translated) Chiplet, a I/O Chiplet or another I/O Controller Chiplet.

I_{XTPKB} An overview of the interfaces between the two chiplets is depicted in Figure 5.12.

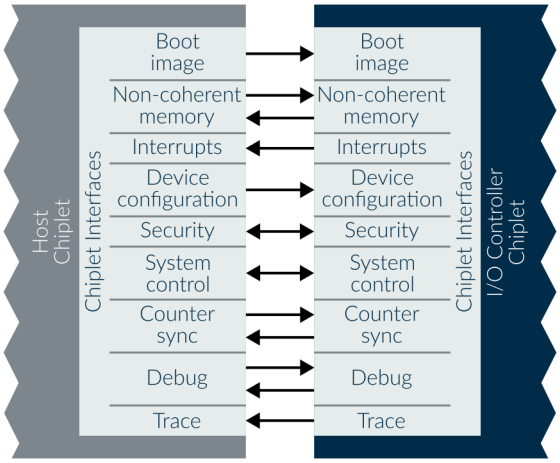

Figure 5.12: I/O Controller Chiplet Interfaces

5.12.1 Boot Image Load Interface

I_{JBQQV} Non-volatile memory can be accessed by a host chiplet. A Boot Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{RDVNB} There is a Boot Image Load Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that is capable of transporting boot images from a host chiplet to memory locations in a I/O Controller Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{CDZVL} A Boot Image Load Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is able to operate in the boot sequence before it can rely on any software processing on the receiving chiplet.

(LEVEL-0)

R_{JHJVP} A Boot Image Load Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{FWTNS} For example, a Boot Image Load Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Inactive* and *Power saving* chiplet activity states.

I_{FNCRV} For implementation details regarding this interface see Boot Image Load Interface.

5.12.2 Non-Coherent Memory Traffic Interface

I_{YRMRK} A Non-Coherent Memory Traffic Interface allows agents in a host chiplet to access system addressable memory in a I/O Controller Chiplet that is directly connected, and in chiplets connected to that chiplet.

I_{HNKGV} A Non-Coherent Memory Traffic Interface allows agents in a I/O Controller Chiplet to access system addressable memory in a host chiplet that is directly connected, and in chiplets connected to that chiplet.

R_{DDDSW} There is a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that transports non-coherent memory traffic in both directions.

(LEVEL-0)

R_{YZHWP} A Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{QFCFD} For example, a Non-Coherent Memory Traffic Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{NXZRC} For implementation details regarding this interface see Non-Coherent Memory Traffic Interface.

5.12.3 Interrupt Handling

I_{CZKKR} This section concerns interrupts that are targeted at application PEs running the main operating system.

5.12.3.1 Interrupts

I_{VNLNT} A I/O Controller Chiplet might contain components that generate interrupts.

R_{YJGHH} There is a Interrupt Interface between a host chiplet and a I/O Controller Chiplet that transports interrupts, as MSIs, from a I/O Controller Chiplet through an MMU to a GIC ITS [4], both in a host chiplet.

(LEVEL-0)

R_{LSJMB} A Interrupt Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{VWNRD} For example, a Interrupt Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{KFHHG} For implementation details regarding this interface see Interrupts.

5.12.4 Device Configuration Interface

R_{PCWSQ} There is a Device Configuration Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that allows host software to configure the device(s) in a I/O Controller Chiplet.

(LEVEL-0)

R_{ZQVFY} A Device Configuration Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{KQWJV} For example, a Device Configuration Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{RFLNG} For implementation details regarding this interface see Device Configuration Interface.

5.12.5 Security

R_{YRHHX} There is a Trusted Security Agent Communication Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that transports messages between the trusted security agents in the chiplets.

(LEVEL-0)

R_{HLXNC} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is able to support the delegation of security lifecycle management and other secure communication between trusted security agents.

(LEVEL-0)

R_{LKCWK} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is protected with encryption, has integrity protection, and has replay protection.

(LEVEL-0)

R_{XZPFB} A Trusted Security Agent Communication Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{MYXRS} For example, a Trusted Security Agent Communication Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{RPXCH} For implementation details regarding this interface see Trusted Security Agent Communication Interface.

5.12.6 System Control Communication

R_{SXXYP} There is a System Control Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{JSHJH} A System Control Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{NHXWC} For example, a System Control Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{QVVYJ} For implementation details regarding this interface see System Controller Communication.

5.12.7 System Counter Synchronization

R_{WBVWT} Where a I/O Controller Chiplet contains a system counter, there is a System Counter Synchronization Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that is capable of synchronizing the system counters in the chiplets.

(LEVEL-0)

R_{SBNMJ} A System Counter Synchronization Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DNVPD} For example, a System Counter Synchronization Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Initial boot* and *Operational* chiplet activity states and be unavailable in the *Secure boot*, *Power saving* and *Inactive* chiplet activity states.

I_{TBYGV} For implementation details regarding this interface see System Counter Synchronization Interface.

5.12.8 Debug

5.12.8.1 Debug Access

R_{RNDMS} There is a Debug Access Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that facilitates access between debug components in the two chiplets.

(LEVEL-0)

I_{LRVTY} For implementation details regarding this interface see Debug Access Interface.

5.12.8.2 Debug Cross-Trigger

R_{TQCXK} There is a Debug Cross-Trigger Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that facilitates debug cross-triggering between the two chiplets.

(LEVEL-0)

R_{QQNKF} A Debug Cross-Trigger Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{DQPNB} For example, a Debug Cross-Trigger Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Secure boot*, *Initial boot*, *Operational* and *Power saving* chiplet activity states and be unavailable in the *Inactive* chiplet activity state.

I_{QSKVY} For implementation details regarding this interface see Debug Cross-Trigger Interface.

5.12.9 Trace

R_{JRYTT} Where a I/O Controller Chiplet contains a CoreSight trace source, there is a Trace Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that is capable of transporting trace data from a CoreSight *Embedded Trace Router* (ETR) [8] contained in a I/O Controller Chiplet to memory locations in a host chiplet, or in other chiplets connected to that chiplet.

(LEVEL-0)

R_{NPCXN} A Trace Interface between a host chiplet and a I/O Controller Chiplet that is directly connected is available for use when the two chiplets share a common chiplet activity state in which the functions associated with the interface are operative.

(LEVEL-0)

U_{HNLHB} For example, a Trace Interface between a host chiplet and a I/O Controller Chiplet that is directly connected might be available in the *Operational* chiplet activity state and be unavailable in the *Inactive*, *Secure boot*, *Initial boot* and *Power saving* chiplet activity states.

I_{TDTWR} For implementation details regarding this interface see Trace Interface.

Chapter 6
Common Chiplet Components

Some chiplet components are common to the specifications of several CSA chiplet types. These components are described in this chapter.

The components are:

- System Controller.
- System Control Agent.
- Trusted Security Agent.
- System Counter.

6.1 System Control

6.1.1 System Controller

D_{XJKRK} The System Controller is a component used for low-level SoC specific management tasks such as power and thermal management.

The system control functionality is distributed over multiple chiplets, typically with a system controller component in each chiplet. One of the system controllers is a primary system controller, the others are secondary system controllers. The primary system controller coordinates the actions of the secondary system controllers.

R_{TMGXQ} There is only one *primary* system controller in the system.

(LEVEL-0)

R_{NHBRK} The primary system controller coordinates the actions of the secondary system controllers.

(LEVEL-0)

D_{BTCVH} A *secondary system controller* is responsible, where applicable, for:

- Chiplet resets.
- Chiplet initialization.
- Chiplet high-level clock control.
- Chiplet performance management.
- Chiplet power mode transitions.
- Chiplet power budget management.
- Chiplet temperature management.
- Chiplet telemetry and sensor management.
- Chiplet watchdog events.

D_{CDNCB} A *primary system controller*, in addition to the responsibilities of a secondary system controller, is responsible, where applicable, for:

- System initialization.
- System power mode coordination.
- System power budget management.
- System temperature management.
- System sensor and telemetry coordination.
- System watchdog events.

I_{SQYMV} Several of these functions require communication between system controllers, for example, to synchronize operations during initialization, to receive budgets, or negotiate power modes and other functions.

R_{CSHRH} There is a communication channel between the primary system controller and each of the secondary system controllers. Where this channel is between two chiplets a System Control Interface is used.

(LEVEL-0)

I_{RJCQQ} There is an initial phase of boot of a system controller where it operates independently, from local boot code.

R_{BZMMM} The communication channel between the primary system controller and a secondary system controller is established during the initial phase of boot of a system controller.

(LEVEL-FULL)

R_{WRJNW} A primary system controller might transfer a boot code image to a secondary system controller that is in another chiplet using a Boot Image Load Interface.

(LEVEL-0)

6.1.1.1 System Controller Interrupts

I_{RPFDL} A system controller receives interrupts to inform it of events occurring in the system, messages from other controller entities or PEs, and actions required by other components involved in the management of the system.

System controller interrupts come directly to the system controller. They do not come through the Arm Generic Interrupt Controller (GIC) infrastructure in the same way as interrupts targeted at Application PEs in the system, but come from the interrupt source directly to the system controller.

If system controller interrupts are required to be sent to a system controller on another chiplet, they are routed to the system controller on the current chiplet. That system controller forwards them as one or more messages on the system control interface to the relevant system controller.

R_{NQZFH} A system controller receives interrupts directly from the source of the interrupt.

(LEVEL-0)

R_{FLDNS} System controller interrupts targeted at a system controller on other chiplets are sent to the system controller on the current chiplet, then forwarded to the target system controller as message(s) on the system control interface.

(LEVEL-0)

6.1.2 System Control Agent

D_{XDMMS} A *system control agent* is a component that supports a system controller in effecting system control policy decisions. A system control agent is fixed function, acting on hardware-level system controls in response to policy settings and direct requests from a system controller, such as safely transitioning power modes, or changing clock gating arrangements.

R_{GGJPQ} The configuration of a system control agent is accessible by software through the system address map with the highest implemented security access level supported by the HW.

(LEVEL-FULL)

U_{JSBNS} A system control agent might be an Arm Power Policy Unit (PPU) [11].

6.2 Trusted Security Agent

I_{GNQVC} The system offers specific security guarantees, according to the requirements of the particular system design. These guarantees are supported by a *security platform*, a collective term that identifies all the hardware and firmware components involved in the delivery of the security guarantees.

Certain components of the security platform are hardware enforced. The hardware enforced components of the security platform are collectively referred to as the system *Hardware Enforced Security* (HES) functionality. They include:

- Security platform boot state.

  The security platform boot state represents the security configurations of the components of the security platform, and measurements and identification of all security platform firmware.

- Security platform attestation.

  Security platform attestation is a feature of complex devices with multiple security agents. Each security agent might implement its own secure boot process, but does not itself provide a security boot state of the platform as a whole. Security platform attestation consists of a measurement of the critical boot stages of all platform security components, that provides an overall platform security attestation.

- Hardware security lifecycle state.

  The security platform might be in a fully trusted state or in a state with some lesser degree of trustworthiness. The *security lifecycle* reflects the transition of the system through these various states as the system experiences its general lifecycle, from manufacture through use to decommissioning.

  The hardware security lifecycle state is enforced by:

  – The maintenance of hardware provisioned parameters, a set of secrets and public identifiers that might be used in determining the trustworthiness of the platform, and to protect security platform assets.
  – Moderating invasive subsystems. For example, ensuring that all debug interfaces have been closed.
  – Ensuring that only authorized security platform firmware can be loaded, and then requiring that no part of the security platform can boot in a non-secure state without authorization from a hardware enforced source.

- Certain cryptographic services that are necessary to security platform attestation, such as the signing of measurements and the derivation of keys.

- Shielded locations such as tamper resistant storage and the encryption of external memory.

For more information on HES, refer to the Arm CCA security model [5].

Refer to Chiplet System Threat Model and Security Goals for further information on the security lifecycle state.

D_{PQHNP} HES functionality is hosted by one or more *Trusted Security Agents*, isolated hardware subsystems dedicated to security. The capabilities of a trusted security agent might vary, from providing secure storage of a unique identifier to hosting multiple security platform functions that require a private execution environment, its own trusted firmware and its own boot process.

A trusted security agent might be responsible for its own security lifecycle management and represent a Root of Trust (RoT) in its own right. A trusted security agent is attestable, so might delegate its security lifecycle management to another trusted security agent.

I_{VDGJB} A trusted security agent is a host to system HES functionality. In a system design composed using chiplets, HES functionality is distributed over multiple chiplets, with one or more trusted security agents in each chiplet that requires or provides any component of system HES functionality.

R_{FFKWC} A single trusted security agent in the system is designated as the primary trusted security agent that acts as the Root of Trust (RoT), the other trusted security agents are designated as secondary trusted security agents.
(LEVEL-0)

R_{HRPTP} The primary trusted security agent accepts the responsibility for the security lifecycle management of all the other trusted security agents in the system.

(LEVEL-FULL)

R_{NPXGM} A secondary trusted security agent delegates its security lifecycle management to the primary trusted security agent.

(LEVEL-FULL)

R_{MWLGW} There is a communication channel between the primary trusted security agent and each of the secondary trusted security agents. Where this channel is between two chiplets a Trusted Security Agent Communication Interface is used.

(LEVEL-0)

I_{RYRZM} There is an initial phase of boot of a trusted security agent where it operates independently, from local boot code or a default state.

R_{VCTWQ} The communication channel between the primary trusted security agent and a secondary trusted security agent is established during the initial phase of boot of a trusted security agent.

(LEVEL-FULL)

R_{LDZYP} A primary trusted security agent might transfer a boot code image to a secondary trusted security agent that is in another chiplet using a Boot Image Load Interface.

(LEVEL-0)

R_{PZFPH} The initial boot of a trusted security agent in a chiplet takes precedence over the initial boot of a system controller in the same chiplet. A trusted security agent has control over when the initial boot of a system controller in the same chiplet starts.

(LEVEL-FULL)

6.2.0.1 Trusted Security Agent Interrupts

I_{JRXKG} A trusted security agent receives interrupts to inform it of events occurring in the system, messages from other controller entities or PEs, and actions required by other components involved in the security management of the system.

Trusted security agent interrupts come directly to the system controller. They do not come through the Arm Generic Interrupt Controller (GIC) infrastructure in the same way as interrupts targeted at Application PEs in the system, but come from the interrupt source directly to the trusted security agent.

If trusted security agent interrupts are required to be sent to a trusted security agent on another chiplet, they are routed to the trusted security agent on the current chiplet. That trusted security agent forwards them as messages on the Trusted Security Agent Communication Interface to the relevant trusted security agent.

R_{RCPHY} A trusted security agent receives interrupts directly from the source of the interrupt.

(LEVEL-0)

R_{JMCGT} Trusted security agent interrupts targeted at a trusted security agent on other chiplets are sent to the trusted security agent on the current chiplet, then forwarded to the target trusted security agent as message(s) on the Trusted Security Agent Communication Interface.

(LEVEL-0)

6.3 System Counter

D_{BTZJT} The system counter is specified by the Arm ARM [2] as a sub-component of the *Generic Timer*, a necessary component in an Arm system. It measures the passing of time in real-time, providing a uniform view of system time to all components in the Arm system that require it. This includes, in addition to the Generic Timer, trace timestamps and RAS telemetry.

D_{TDDHD} In a system that is composed of chiplets the system counter is distributed over multiple chiplets. There is a primary system counter in a single chiplet that provides the source of the count, and secondary system counters in all other chiplets that require a view of system time. These secondary system counters are used for local distribution of the system count within their respective chiplet.

R_{QVVFX} A system counter meets the requirements as specified by the Arm ARM [2] *Generic Timer* and by the Arm Base System Architecture [1] *Clock and Timer Subsystem*.

(LEVEL-0)

R_{SZQTS} There is only one *primary* system counter in the system.

(LEVEL-0)

6.3.1 System Counter Synchronization

R_{NQWTT} A secondary system counter is synchronized to the primary system counter. Synchronization might be effected directly between a secondary system counter and the primary system counter, or indirectly from one secondary system counter to another, forming a sequence back to the primary system counter.

(LEVEL-0)

R_{TQYQP} Where a secondary system counter is synchronized to another system counter, the entire counter-to-counter synchronization sequence back to the primary system counter contains no loops.

(LEVEL-0)

I_{CQLWT} A secondary system counter might request a synchronization to another system counter to, for example, restore or update it following exit from a low power mode, or to maintain a regular update cadence.

R_{YGTLQ} Where a secondary system counter is synchronized to another system counter, the protocol used for the synchronization supports the initiation of the synchronization process by the system counter that has the greater number of steps in the synchronization sequence back to the primary system counter.

(LEVEL-0)

R_{KCSBB} The protocol used for the synchronization of the system counters operates so that any agent that reads the system counter during communication with another agent does not observe the count going backwards. This aligns with the rules in the Arm ARM [2].

(LEVEL-0)

Chapter 7
Chiplet Interface Implementation

I_{GMDKN} This chapter describes the mapping of interfaces described in the chiplet type descriptions to the protocols and transport mechanisms used in a chiplet implementation.

I_{VMGWH} This chapter specifies the interfaces in terms of the available standards and architecture. The CSA does not define any new interface standards. Where applicable standards are not available for an interface the transport and protocol are IMPLEMENTATION DEFINED.

I_{KPDLW} The protocols and transports that the CSA specifies for interfaces might be versioned. A chiplet can only be used in a system when every interface with another chiplet has either:

- the same protocol and transport version.
- different protocol and transport versions where compatibility between the versions is supported.

When planning for the reuse of a chiplet, the system or chiplet designer must have awareness of the road-map for each protocol and transport that is under consideration, and plans for backwards compatibility between the versions used.

U_{QLGCD} Multiple interfaces might share a single transport when the transport requirements of each interface are common, and provided that when the transport is shared the requirements of all the participating interfaces are satisfied. Where an interface has such a requirement, a note is provided.

I_{GNDXP} The sharing of a transport by multiple interfaces might change the performance of those interfaces, for example, it might affect latency, transaction ordering, and deadlock avoidance mechanisms.

U_{WJVQV} An interface might be implemented as a service provided by a system component that has its own interface between chiplets. Where an interface might be implemented this way, a note is provided.

7.1 Hardware Fault Requirements for Chiplet Interfaces

I_{DQSRJ} Systems that have functional safety, reliability, availability, serviceability or similar requirements might employ mechanisms that detect, measure, report, correct or repair transient or permanent hardware faults in chiplet interfaces.

U_{KHZLN} The implementation of an interface, as specified by the CSA, meets any system-level requirements of that interface that concerns hardware faults, where those requirements are provided by a chiplet interface fault model specific to that implementation.

U_{STTGM} The specific protocol or transport of an interface, as specified by the CSA, might include features that support meeting system-level requirements that concern hardware faults. This includes interfaces that use the UCIe [12] and I3C [13] transport mechanisms. These features are considered as part of defining any IMPLEMENTATION DEFINED functionality of the interface.

7.2 Interface Protocols

This section describes the interface protocols for the specified interfaces.

7.2.1 Boot Image Load Interface

I_{FXRVR} Boot images for certain components in the system are loaded into chiplet memory before any software communication protocols are active.

R_{KJVCD} The protocol for the Boot Image Load Interface is IMPLEMENTATION DEFINED.

(LEVEL-FULL)

U_{PDHSQ} The Boot Image Load Interface might be implemented as a service that is provided by a system component such as a trusted security agent or a system controller.

7.2.2 Coherent Memory Traffic Interface

I_{SBKRM} A Coherent Memory Traffic Interface between two chiplets allows coherent agents in either chiplet to access system addressable memory and caches in other chiplets in the system, and to transport the relevant snoop and DVM messages.

R_{JCLGK} The protocol for the Coherent Memory Traffic Interface is AMBA CHI Chip-to-Chip (CHI C2C) [14].

(LEVEL-1)

I_{CVWBV} The AMBA CHI Chip-to-Chip (CHI C2C) specification [14] includes a table "Expectations when exchanged C2C specific properties differ". The table includes an item that concerns the situation where the transmitter is using more *Resource Planes* than the receiver is able to support.

R_{BBMHB} A transmitter chiplet that sends messages using a AMBA CHI Chip-to-Chip (CHI C2C) [14] for the Coherent Memory Traffic Interface to a receiver chiplet with fewer *Resource Planes* is able to consolidate the traffic onto the reduced number of *Resource Planes*. There is no combination of messages of different classes that, with the unavoidable reduction of traffic dependency isolation that results from this consolidation, carries a risk of violating forward progress guarantees or service time guarantees.

(LEVEL-1)

R_{PPCHS} Each chiplet that is using the Coherent Memory Traffic Interface implements a minimum of AMBA CHI version G [3].

(LEVEL-1)

7.2.3 Untranslated I/O Coherent Memory Traffic Interface

I_{MVZPN} A Untranslated I/O Coherent Memory Traffic Interface allows I/O coherent agents in a chiplet to access system addressable memory and caches in other chiplets in the system.

R_{DPWJP} The protocol for the Untranslated I/O Coherent Memory Traffic Interface is IMPLEMENTATION DEFINED.

(LEVEL-FULL)

7.2.4 I/O Coherent Memory Traffic Interface

I_{JTWJV} A I/O Coherent Memory Traffic Interface allows I/O coherent agents in a chiplet to access system addressable memory and caches in other chiplets in the system.

R_{ZKQNK} The protocol for the I/O Coherent Memory Traffic Interface is AMBA CHI Chip-to-Chip (CHI-C2C) [14].
(LEVEL-1)

I_{NCHSD} The AMBA CHI Chip-to-Chip (CHI C2C) specification [14] includes a table "Expectations when exchanged C2C specific properties differ". The table includes an item that concerns the situation where the transmitter is using more *Resource Planes* than the receiver is able to support.

R_{RGCVB} A transmitter chiplet that sends messages using a AMBA CHI Chip-to-Chip (CHI C2C) [14] for the I/O Coherent Memory Traffic Interface to a receiver chiplet with fewer *Resource Planes* is able to consolidate the traffic onto the reduced number of *Resource Planes*. There is no combination of messages of different classes that, with the unavoidable reduction of traffic dependency isolation that results from this consolidation, carries a risk of violating forward progress guarantees or service time guarantees.
(LEVEL-1)

R_{DJWHS} Each chiplet that is using the I/O Coherent Memory Traffic Interface implements a minimum of CHI version G [3].
(LEVEL-1)

7.2.5 Non-Coherent Memory Traffic Interface

I_{RGHYT} A Non-Coherent Memory Traffic Interface between two chiplets allows agents in either chiplet to access system addressable memory in other chiplets in the system.

R_{WVPQS} The protocol for the Non-Coherent Memory Traffic Interface is AMBA CHI Chip-to-Chip (CHI C2C) [14].
(LEVEL-1)

I_{RZVQY} The AMBA CHI Chip-to-Chip (CHI C2C) specification [14] includes a table "Expectations when exchanged C2C specific properties differ". The table includes an item that concerns the situation where the transmitter is using more *Resource Planes* than the receiver is able to support.

R_{KKKJS} A transmitter chiplet that sends messages using a AMBA CHI Chip-to-Chip (CHI C2C) [14] for the Non-Coherent Memory Traffic Interface to a receiver chiplet with fewer *Resource Planes* is able to consolidate the traffic onto the reduced number of *Resource Planes*. There is no combination of messages of different classes that, with the unavoidable reduction of traffic dependency isolation that results from this consolidation, carries a risk of violating forward progress guarantees or service time guarantees.
(LEVEL-1)

R_{QYJFV} Each chiplet that is using the Non-Coherent Memory Traffic Interface implements a minimum of AMBA CHI version G [3].
(LEVEL-1)

7.2.6 Device Configuration Interface

I_{JXTCR} A Device Configuration Interface allows host software to configure a device that is in an expansion chiplet through a memory mapped interface.

7.2.6.1 Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet

R_{TZRTL} The protocol for the Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet is AMBA CHI Chip-to-Chip (CHI-C2C) [14].

(LEVEL-1)

I_{GNPYY} The AMBA CHI Chip-to-Chip (CHI C2C) specification [14] includes a table "Expectations when exchanged C2C specific properties differ". The table includes an item that concerns the situation where the transmitter is using more *Resource Planes* than the receiver is able to support.

R_{CYJBK} A transmitter chiplet that sends messages using a AMBA CHI Chip-to-Chip (CHI C2C) [14] for the Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet to a receiver chiplet with fewer *Resource Planes* is able to consolidate the traffic onto the reduced number of *Resource Planes*. There is no combination of messages of different classes that, with the unavoidable reduction of traffic dependency isolation that results from this consolidation, carries a risk of violating forward progress guarantees or service time guarantees.

(LEVEL-1)

R_{VHMXL} Each chiplet that is using the Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet implements a minimum of AMBA CHI version G [3].

(LEVEL-1)

7.2.6.2 Device Configuration Interface for a I/O Coherent Expansion (Untranslated) Chiplet

R_{TQRTF} The protocol for the Device Configuration Interface for a I/O Coherent Expansion (Untranslated) Chiplet is IMPLEMENTATION DEFINED.

(LEVEL-FULL)

7.2.6.3 Device Configuration Interface for a I/O or a Fully Coherent Expansion (Remote Translation) Chiplet

R_{DDXMP} The protocol for the Device Configuration Interface for a I/O Coherent Expansion (Remote Translation) Chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet is PCIe [15].

(LEVEL-FULL)

7.2.7 Address Translation Interface

I_{JNHGF} A Address Translation Interface allows a device in a chiplet to request address translations from an MMU component in another chiplet that is directly connected.

R_{VLQZG} The protocol for the Address Translation Interface is PCIe ATS [15].

(LEVEL-FULL)

7.2.8 Interrupt Handling

I_{YFJMM} This section concerns interrupts that are targeted at application PEs running the main operating system.

7.2.8.1 Interrupts

I_{RJNQG} Interrupts are sent as either MSI or MSI-64 messages, or using memory-mapped accesses between GICD components.

R_{XBWSM} For chiplets that contain a GICD, interrupts from the chiplet are sent to the GICD and then the GICD Communication Interface is used to forward the interrupt to the target GICD if required.
(LEVEL-FULL)

R_{XLNDW} For chiplets that do not contain a GICD, if they contain an MMU, then interrupts are sent as MSI-64.
(LEVEL-FULL)

R_{NKRPZ} For chiplets that do not contain a GICD, if they do not contain an MMU, then interrupts are sent as MSI.
(LEVEL-FULL)

7.2.8.2 GICD Communication

R_{NPVYB} The protocol for communication of GIC distributors (GICD) in different chiplets is IMPLEMENTATION DEFINED.
(LEVEL-FULL)

I_{JSJQY} The method for GICD Communication is specific to a GIC implementation. The Arm GIC-600 [16] and GIC-700 [6] Interrupt Controllers have proprietary solutions.

7.2.9 Trusted Security Agent Communication Interface

I_{WYXND} The trusted security agent in each chiplet is responsible for hardware enforced security, and might provide higher-level security functionality. System level hardware enforced security is distributed over these trusted security agents. All secondary trusted security agents in the system delegate their security lifecycle management to the primary trusted security agent.

I_{BTYND} The security of the communication between trusted security agents relies upon the use of a dedicated interface. Additional interface functionality is only implemented on the same interface when provided directly as a service of the trusted security agent.

R_{TDLVS} The protocol for the Trusted Security Agent Communication Interface is IMPLEMENTATION DEFINED.
(LEVEL-FULL)

U_{XDGJH} If the Boot Image Load Interface is implemented as a service of the trusted security agent then the Trusted Security Agent Communication Interface is shared with this function.

7.2.10 System Control Interface

I_{CLDTS} The system controllers and system control agents in the chiplets are responsible for various system control and power management functions. Several of these functions require communication between controllers.

R_{PXBTX} The protocol for the System Control Interface is the System Control and Management Interface (SCMI) [17].
(LEVEL-FULL)

7.2.11 System Counter Synchronization Interface

R_{QHGSY} The protocol for synchronization of system counter values is IMPLEMENTATION DEFINED.
(LEVEL-FULL)

I_{MCWRJ} See the definition of the system counter synchronization for more protocol requirements.

7.2.12 Debug

7.2.12.1 Debug Access Interface

R_{LXCQZ} Debug access protocol to a chiplet is provided by JTAG IEEE Std 1149.

(LEVEL-FULL)

7.2.12.2 Debug Cross-Trigger Interface

I_{SSPTF} See Debug Cross-Trigger Interface Transport.

7.2.13 Trace Interface

I_{TQVNQ} A CoreSight *Embedded Trace Router* (ETR) [8] contained in a chiplet can write trace data to system memory.

I_{KDGYH} See Trace Interface transport.

7.3 Interface Transports

This section describes the interface transports for the specified interfaces.

7.3.1 Boot Image Load Interface

I_{LHQXN} Boot images for certain components in the system are loaded into chiplet memory before any software communication protocols are active.

R_{GJBYZ} The transport for the Boot Image Load Interface is IMPLEMENTATION DEFINED, but includes supporting hardware for converting transport transactions to memory mapped writes.

(LEVEL-FULL)

U_{GDGJF} A recommended transport for the Boot Image Load Interface is an I3C interface [13] with supporting hardware for converting I3C transactions to memory mapped writes.

X_{TWFSV} The I3C interface [13] is a standard, simple interface that is a fit for the purpose of transporting the Boot Image Load Interface.

U_{GSFJN} The Boot Image Load Interface might be implemented as a service that is provided by a system component such as a trusted security agent or a system controller.

7.3.2 Coherent Memory Traffic Interface

I_{QQWYS} A Coherent Memory Traffic Interface between two chiplets allows coherent agents in either chiplet to access system addressable memory and caches in other chiplets in the system, and to transport the relevant snoop and DVM messages.

R_{ZZXCL} The transport for the Coherent Memory Traffic Interface is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

U_{GDKQF} To use the UCIe interface as a transport, each chiplet converts between the AMBA CHI protocol and AMBA CHI C2C protocol, which includes support for UCIe.

7.3.3 Untranslated I/O Coherent Memory Traffic Interface

I_{BZSFF} A Untranslated I/O Coherent Memory Traffic Interface allows I/O coherent agents in a chiplet to access system addressable memory and caches in other chiplets in the system.

R_{QDDHW} The transport for the Untranslated I/O Coherent Memory Traffic Interface is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

7.3.4 I/O Coherent Memory Traffic Interface

I_{RKZPK} A I/O Coherent Memory Traffic Interface allows I/O coherent agents in a chiplet to access system addressable memory and caches in other chiplets in the system.

R_{NCZJN} The transport for the I/O Coherent Memory Traffic Interface is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

U_{SMPWS} To use the UCIe interface as a transport, each chiplet converts between the AMBA CHI protocol and AMBA CHI C2C protocol, which includes support for UCIe.

7.3.5 Non-Coherent Memory Traffic Interface

I_{JHFWG} A Non-Coherent Memory Traffic Interface between two chiplets allows agents in either chiplet to access system addressable memory in other chiplets in the system.

R_{BCQKB} The transport for the Non-Coherent Memory Traffic Interface is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

7.3.6 Device Configuration Interface

I_{KBLTT} A Device Configuration Interface allows host software to configure a device that is in an expansion chiplet through a memory mapped interface.

7.3.6.1 Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet

R_{YNCSL} The transport for the Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

7.3.6.2 Device Configuration Interface for a I/O Coherent Expansion (Untranslated) Chiplet

R_{JDCPH} The transport for the Device Configuration Interface for a I/O Coherent Expansion (Untranslated) Chiplet is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

7.3.6.3 Device Configuration Interface for a I/O or a Fully Coherent Expansion (Remote Translation) Chiplet

R_{CJBHK} The transport for the Device Configuration Interface for a I/O Coherent Expansion (Remote Translation) Chiplet is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

7.3.7 Address Translation Interface

I_{NBQFR} A Address Translation Interface allows a device in a chiplet to request address translations from an MMU component in another chiplet that is directly connected.

R_{CTJXM} The transport for the Address Translation Interface is the UCIe v1.1 [12] streaming protocol.

(LEVEL-FULL)

7.3.8 Interrupt Handling

I_{CCVFN} This section concerns interrupts that are targeted at application PEs running the main operating system.

7.3.8.1 Interrupts

I_{ZRGXC} Interrupts are sent as either MSI or MSI-64 messages, or using memory-mapped accesses between GICD components.

R_{BGHPZ} Interrupts are transported using one of the following in priority order:

1. Coherent Memory Traffic Interface
2. I/O Coherent Memory Traffic Interface
3. Non-Coherent Memory Traffic Interface

(LEVEL-FULL)

7.3.9 Trusted Security Agent Communication Interface

I_{DRPRG} The trusted security agent in each chiplet is responsible for hardware enforced security, and might provide higher-level security functionality. System level hardware enforced security is distributed over these trusted security agents. All secondary trusted security agents in the system delegate their security lifecycle management to the primary trusted security agent.

I_{TYLFZ} The security of the communication between trusted security agents relies upon the use of a dedicated interface. Additional interface functionality is only implemented on the same interface when provided directly as a service of the trusted security agent.

R_{LQPGR} The transport for the Trusted Security Agent Communication Interface is IMPLEMENTATION DEFINED.
(LEVEL-FULL)

U_{DNJVQ} A recommended transport for the Trusted Security Agent Communication Interface is a dedicated I3C interface [13].

X_{LPGMG} The I3C interface [13] is a standard, simple interface that is a fit for the purpose of transporting the Trusted Security Agent Communication Interface.

U_{PXGQL} If the Boot Image Load Interface is implemented as a service of the trusted security agent then the Trusted Security Agent Communication Interface is shared with this function.

7.3.10 System Control Interface

I_{SJSXS} The system controllers and system control agents in the chiplets are responsible for various system control and power management functions. Several of these functions require communication between controllers.

I_{FKMPZ} There are multiple transports between chiplets that are capable of carrying the communication between system controllers and system control agents. The particular transport used depends on the chiplet activity state and the associated availability of the interfaces.

R_{XYZYJ} When two directly connected chiplets are both in the *Operational* chiplet activity state the Coherent Memory Traffic Interface is available and is used as the transport for the System Control Interface.
(LEVEL-FULL)

U_{TSQPW} When using the Coherent Memory Traffic Interface to transport the System Control Interface it is expected the SCMI [17] protocol communicates by sending messages to a Message Handling Unit (MHU) on the other chiplet.

R_{GTDWN} When either or both of two directly connected chiplets are in one of the *Secure boot*, *Initial boot* or *Power saving* chiplet activity states, the transport for the System Control Interface is IMPLEMENTATION DEFINED.
(LEVEL-FULL)

U_{MDFPH} A recommended transport for the System Control Interface, when either or both of two directly connected chiplets are in one of the *Secure boot*, *Initial boot* or *Power saving* chiplet activity states, is a dedicated I3C interface [13].

X_{ZPTVP} The I3C interface [13] is a standard, simple interface that is a fit for the purpose of transporting the System Control Interface.

U_{NLWKD} If the Boot Image Load Interface is implemented as a service of the system controller then the System Control Interface is shared with this function.

U_{YNTTZ} A dedicated **WAKEUP** wire signal might be required between chiplets where a connected chiplet can enter a power mode in which it is unresponsive to SCMI communication, such as the *Power saving* chiplet activity state.

7.3.11 System Counter Synchronization Interface

R_{YLBWC} The transport for synchronization of system counter values is IMPLEMENTATION DEFINED.
(LEVEL-FULL)

I_{TLDNK} See the definition of the system counter synchronization for more protocol requirements.

7.3.12 Debug

7.3.12.1 Debug Access Interface

R_{HKNHZ} Debug access transport is provided by a JTAG Test Access Port (TAP).

(LEVEL-FULL)

U_{BGJJT} Debug access to the chiplets in a system is provided by the JTAG Test Access Port (TAP) of each chiplet. Each port might be accessed individually, or the ports of multiple chiplets might be connected in the manner of a daisy-chain. In the daisy-chain structure common TCK and TMS signals are provided to all chiplets in the chain, and the TDI and TDO signals are daisy-chained through each chiplet, so that the TDO of the first chiplet is fed to the TDI of the next until all chiplets in the chain are connected.

U_{BWDZW} Multiple chiplets in the system might have a shared address map, such that a Test Access Port (TAP) in a single chiplet can access the debug components of other chiplets using memory-mapped transactions.

U_{ZRQNY} Although the debug components of a chiplet might be accessible by a TAP in another chiplet in the system via memory-mapped transactions, the ability to access each chiplet separately allows access when a Coherent Memory Traffic Interface or other inter-chiplet memory access interface might not be available.

7.3.12.2 Debug Cross-Trigger Interface

R_{VFMQX} A CoreSight *Channel Interface* (see [7]), is used for communication of triggers between CoreSight *Cross Trigger Interface* (CTI) components in different chiplets. The interface has one channel in each direction and the asynchronous form is used. The interface requires four signals.

(LEVEL-FULL)

U_{RBFZG} If multiple events are presented to a single channel interface in close succession, more rapidly than the receiver can acknowledge them, they might be interpreted as a single event.

U_{FVWMC} The system debugger is provided with information about which CTI triggers in a chiplet are connected to another chiplet. This information is used to set up the trigger system without a loop that spans multiple chiplets.

7.3.13 Trace

I_{JSTPJ} A CoreSight *Embedded Trace Router* (ETR) [8] contained in a chiplet can write trace data to system memory.

R_{TWFXJ} Trace data from a CoreSight *Embedded Trace Router* (ETR) [8] is transported between chiplets, where available, using the Coherent Memory Traffic Interface, I/O Coherent Memory Traffic Interface or Non-Coherent Memory Traffic Interface.

(LEVEL-FULL)

Chapter 8
Chiplet System Threat Model and Security Goals

The design of a system that incorporates chiplets follows a security development procedure. The goal of security development is to minimize security vulnerabilities in development and reduce the attack surface of a system throughout the lifecycle of its deployment. See [18] for more information about the Security Development Lifecycle (SDL).

A complete security risk assessment, one that includes threat modelling, risk assessment and mitigation, is beyond the scope of the CSA. Such an assessment requires additional implementation specific information, particularly with regard to the impact of risks. Instead, this chapter is limited to the identification of the threats that are exposed through the specification by the CSA of chiplet types and their interfaces, and the security goals that mitigate them. It supports the designer in performing a comprehensive security risk assessment as part of the security development of a chiplet or a system using chiplets.

8.1 Chiplet System Security Definitions

8.1.1 Chiplet System Lifecycle

A typical chiplet system lifecycle is depicted in Figure 8.1. The various stages are annotated with the stakeholders that are involved or have a direct vested interest in it.

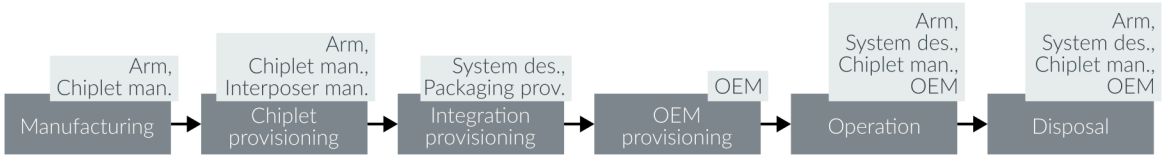

Figure 8.1: Chiplet System Lifecycle & Stakeholders

The following table describes the lifecycle stages in more detail.

Table 8.1: Chiplet System Lifecycle Stages & Stakeholders

Stage	Stakeholders	Description
Manufacturing	Arm, Chiplet manufacturer	The fabrication of the chiplets.
Chiplet provisioning	Arm, Chiplet manufacturer, Interposer manufacturer	Security provisioning of individual chiplets and any interposers.
Integration provisioning	Chiplet system designer, Packaging provider	Security provisioning of chiplets when the chiplets and any interposers are composed into a system and packaged.
OEM provisioning	OEM	Security provisioning of the packaged chiplet system when deployed by the product OEM.
Operation	Arm, Chiplet system designer, Chiplet manufacturer, OEM	Initial boot of the security platform, that is distributed over multiple chiplets, and operation of the system.
Disposal	Arm, Chiplet system designer, Chiplet manufacturer, OEM	Revoking of security provisioning as part of the managed disposal of the chiplet system.

8.1.2 Trust Boundaries

A *trust boundary* is a logical security perimeter that defines areas with different levels of trust. Information or responsibility that crosses a trust boundary is moving between entities that have differing security requirements. An entity could be implemented in software, hardware, or a combination of both and could be inside or outside the system. A trust boundary definition might apply physically to certain components of the system or regions of memory, or temporally according to changes in levels of operational privilege.

In an Arm system certain trust boundaries are defined by the extent of the system. When the Arm system is

constructed using chiplets, trust boundaries can extend over multiple chiplets.

The following trust boundaries can span multiple chiplets, and any interposers or other connections between them:

- The *physical addressing trust boundary*: The use of physical addresses. When, within this trust boundary, address translations and protection checks are made by an MMU, they are trusted to be correct. Memory accesses that originate from outside this trust boundary are untrusted and are not permitted unless subject to a check that occurs within the trust boundary.
- The *Secure memory trust boundary*: The isolation of Secure memory regions that are only accessible in the Secure security state, not in other security states including Non-secure. Memory transactions might be marked as Secure. Transactions that are marked as Secure are assumed to be so marked correctly within this trust boundary.
- The *Realm memory trust boundary*: If the FEAT_RME Arm Architecture [2] feature is implemented, the isolation of Realm memory regions that are only accessible in the Realm security state, not in other states including Non-secure. Memory transactions might be marked as Realm. Transactions that are marked as Realm are assumed to be marked correctly within this trust boundary.
- The *Root memory trust boundary*: If the FEAT_RME Arm Architecture [2] feature is implemented, the isolation of Root memory regions that are only accessible in the Root security state, not in other states including Non-secure. Memory transactions might be marked as Root. Transactions that are marked as Root are assumed to be marked correctly within this trust boundary.

8.1.3 Assumptions & Constraints

Certain assumptions about the system are made, and certain constraints apply, that cannot be changed to mitigate a security risk.

The following assumptions are made when identifying security risks:

- The chiplet types as specified by the CSA are components of an *Arm system* that is constructed using chiplets. All chiplets that are part of the Arm system correspond to CSA chiplet types.
- The system provides a *security platform* (see Trusted Security Agent).
- If a function that is only a part of a chiplet cannot be attested or otherwise loses the trust of the system, then the remaining functionality of the chiplet cannot gain or maintain the trust of the system. The chiplet as a whole is designated as untrusted.
- Individual chiplets are not exposed to a greater risk of physical attack, by virtue of being chiplets, than if they were monolithic chips.
- The interfaces between chiplets are at risk of physical attack. Such interfaces use physical connections through media such as the package substrate, silicon interposers, or Through Silicon Vias (TSVs). The attack can be made during the manufacture of the system, or after manufacturing is complete.

The following constraints apply when identifying security risks:

- All instances of chiplet types in the system are uniquely identified. This identification is used to support the attestation of the individual chiplets.
- The provenance of individual chiplets is not guaranteed to be fully understood and accepted by the system until they are attested. Attestation might be made through certification as a part of the manufacturing process, using measurements made as a part of the system boot process, or a combination of both.

8.1.4 Stakeholders and Assets

A stakeholder is a person or group with a sense of property and volition that places value in the system, or a part of it. The following table lists the stakeholders that are identified in a system constructed using chiplets as specified by the CSA.

Table 8.2: System Security Stakeholders

Stakeholder	Rationale
Arm	As an architecture developer. Any damage to a system designed using chiplets as specified by the CSA can have implications for Arm's reputation and revenue.
Chiplet-system designer	A system is constructed using chiplets as specified by the chiplet-system designer with certain expectations of correct function. Damage to the system can have implications for the designer's reputation and revenue.
Chiplet manufacturer	Damage to a chiplet supplier's product (a chiplet design, or a fabricated chiplet) and consequential damage to a system that it is used in can have implications for the supplier's reputation and revenue.
Interposer manufacturer	Damage to an interposer supplier's product and consequential damage to a system it is used in can have implications for the supplier's reputation and revenue.
Packaging provider	A system is assembled and packaged with certain expectations of integrity and quality. Damage to the system can have implications for the packaging provider's reputation and revenue.
OEM	A system is constructed by the chiplet system integrator at the request of the OEM. The OEM then vouches for the correct function of the system as it is deployed. Damage to the system can have implications for the OEM's reputation and revenue.
End user	The end user entrusts private information to the system and expects the system functionality to be maintained over time. Damage to the system can negatively affect the end user's trust in the system.

The following general classes of assets are identified in a system constructed using chiplets as specified by the CSA:

- Hardware, immutable or exclusive resources of the security platform.
  - Includes information guaranteeing security platform identity, used for attestation.
- Mutable resources (including code and data) of the security platform.
  - Includes software that is part of the security platform, such as trusted firmware.
- Confidentiality and integrity of information that is communicated between chiplets.

8.2 Threat Analysis

8.2.1 Adversarial Models

Adversarial models are descriptions of the behaviors of the adversaries that a designer must consider defenses for when constructing a system. Seven classes of adversarial model are described below: *AM 0*, *AM 1*, *AM 2*, *AM 3*, *AM 4*, *AM 5* and *AM 6*.

AM 0, *AM 1*, *AM 2*, *AM 3*, *AM 4* and *AM 6* are in scope for systems designed using chiplets as specified by the CSA.

AM 0

The adversary is capable only of accessing data that does not require physical access to the system or the ability to run software on it. The adversary is intercepting or providing data or requests to the target system through a network or other remote connection.

The adversary can, for example:

- Read any input and output to and from the system, using external devices.
- Provide, forge, replay or modify such inputs and outputs.
- Measure the timing of the observable operations of the system, either for normal operation or as a response to arranged inputs (such as timing attacks on web servers). The measurement can be used to infer information that is internal to the system.

AM 1

The adversary can mount attacks from software running on the system.

In addition to *AM 0* the adversary can, for example:

- Attempt software exploitation by running malicious software on the system.
- Exploit access to any memory mapped configuration, monitoring infrastructure, or debug infrastructure of the system.
- Make a side channel analysis that relies on software exposed hardware features of the system.
- Perform software induced glitching of resources with techniques such as Rowhammer or Raspberry.

AM 2

The adversary has physical access to the system, but is capable of mounting hardware attacks that are only passive, use reversible modifications and that do not require techniques that are physically invasive of the chiplets. Some communications and state within the system can be observed.

In addition to *AM 1* the adversary can, for example:

- Make a side channel analysis that requires measurement of physical properties (including any leakage source such as electromagnetic emissions, power consumption, photonic emissions, or acoustic channels).
- Connect malicious hardware to the system without permanently modifying the system.
- Gain access to the internal state of the system by inserting interposers between the system and external components, such as memory devices. Such access might be used to observe communications that pass through the interposer.

AM 3

The adversary has physical access to the system and is capable of mounting hardware attacks that are only active, might use irreversible modifications, and that do not require techniques that are physically invasive of the chiplets. Some communications and state within the system can be observed and interfered with.

In addition to *AM 2* the adversary can, for example:

- Connect malicious hardware to the system that might be a permanent modification to the system.

- Gain access to the internal state of the system by inserting interposers between the system and external components, such as memory devices. Such access might be used to read, block, replay, or inject transactions.

AM 4

The adversary applies techniques that are physically invasive of the interconnections between chiplets, and the packaging of the chiplets.

In addition to *AM 3* the adversary can, for example:

- Probe the packaging substrate to make contact with or disrupt physical communication channels between chiplets.
- Probe the silicon interposers to make contact with or disrupt physical communication channels between chiplets.

AM 5

The adversary applies techniques that are physically invasive of the chiplets.

In addition to *AM 4* the adversary can, for example:

- Apply chip decapsulation through laser or chemical etching followed by microphotography to reverse engineer the chiplet.
- Use a focused ion beam to perform gate-level modification.

AM 6

The adversary has access to the supply chain that provides the component chiplets, chiplet-to-chiplet interposers and any integration-time provisioned firmware images that are assembled into the system. Individual chiplets, interposers or firmware images can be substituted or modified.

For example, the adversary can:

- Interfere with the hardware provisioning process of chiplets to disrupt the HES functionality.
- Substitute a tested and *known good* chiplet with one that has failed testing.
- Substitute a chiplet with another that is an impersonation, with an unexpected capability to observe or disrupt some functionality of the system.
- Substitute a chiplet-to-chiplet interposer with another that is an impersonation, with an unexpected capability to observe or disrupt some functionality of the system.
- Substitute a firmware image with another that is an impersonation, with an unexpected capability to observe or disrupt some functionality of the system.

8.2.2 Security Threats and Security Goals

The system designer considers the security threats to an Arm system constructed using chiplets as defined by the CSA.

The following tables summarize the security threats that are a direct result of using chiplets in a system. The system designer is aware that there exists other security threats and vulnerabilities that are not directly related to the use of chiplets, and apply also to a system designed without chiplets. These threats are not included in the following enumeration.

The threats are labelled according to the STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) model.

Spoofing a chiplet identity.	
Description	An adversary impersonates the unique identification of a valid chiplet instance and gains privileged access to the system. For example, an adversary might substitute a chiplet with another one that is recycled or has an unexpected capability to observe or disrupt some functionality of the system.
STRIDE category	Spoofing.
Adversarial model	AM 6.
Security goal	Authenticity. Each chiplet in the system must have a unique identity that cannot be impersonated and is registered together with the current security lifecycle state of the chiplet.
Mitigating action (suggested)	This threat is transferred to the chiplet manufacturer and the system integrator. For example, implement a Physical Unclonable Function (PUF) in the chiplet, with an associated registration and authentication scheme.

Spoofing a chiplet firmware owner's identity.	
Description	An adversary might steal or tamper with a firmware image signing key that is provisioned to a chiplet during manufacture, and use it to sign a malicious firmware image with an unexpected capability to observe or disrupt some functionality of the system. Firmware updates in a system built using chiplets might be more complex than for a monolithic system and this extra complexity potentially increases the attack surface. For example, there might be complex dependencies between chiplets for provisioning and storage of firmware images, and between compatible versions of firmware images.
STRIDE category	Spoofing.
Adversarial model	AM 6.
Security goal	Tamper resilience. Measures for tamper resistance, detection and prevention must be applied to the storage of the firmware image signing key.
Mitigating action (suggested)	This threat is transferred to the chiplet manufacturer and the system integrator. For example, include manufacturing-level obfuscation of the firmware image signing key as it is stored in the chiplet to make it harder for an adversary to access, and include a mechanism that detects tampering attempts.

Chiplet firmware image tampering.	
Description	A chiplet system integrator might source chiplets from multiple suppliers, that are provisioned with firmware that originates from multiple sources. This presents a larger attack surface when compared to a monolithic system design. A firmware image might be tampered with by an adversary in the supply chain, to include an unexpected capability to observe or disrupt some functionality of the system.
STRIDE category	Tampering.
Adversarial model	AM 6.
Security goal	Integrity of the firmware image. The integrity of firmware images must be measured and attested. Only authorized images are executed. A mechanism for the secure update of firmware images ensures the continued integrity of the images as updates are deployed.
Mitigating action (suggested)	This threat is transferred to the chiplet manufacturer and the system integrator. For example, secure boot and secure image loading processes prevent the execution of unauthorized firmware images, or validation of the legitimacy of firmware images before installation can provide a similar protection.

Chiplet firmware image rollback to an earlier version.	
Description	A system built using multiple chiplets might have complex dependencies between the versions of the firmware images that are provisioned to the chiplets during manufacture. Coordination between chiplets of changes to the versions of firmware images might be required. An adversary might exploit the mechanisms for this coordination to leave intact a legitimate, but outdated, version of a firmware image. An outdated firmware image might include a security weakness that has been corrected in a later version. Or, the combination of mis-matched versions of firmware images in the system might expose a weakness that compromises the security of the system.
STRIDE category	Tampering.
Adversarial model	AM 6.
Security goal	Temporal integrity of the firmware image. Images are encrypted when stored in a manner that protects integrity. A system Root of Trust (RoT) uses measurements and makes an attestation of all the firmware images in the system and ensures their versions align with current expectations.
Mitigating action (suggested)	This threat is controlled by employing a process of securely checking the integrity of all firmware images and their expected versions at a system level.

Tampering with a chiplet-to-chiplet interposer design.	
Description	An adversary modifies a chiplet-to-chiplet interposer design during manufacture to insert passive signals or active circuits that are accessed when the system is operational through physical probing of the package. These signals or circuits are used to disrupt the availability of some functionality of the system.
STRIDE category	Tampering.
Adversarial model	AM 6.
Security goal	Integrity of the interposer design.
Mitigating action (suggested)	This threat is transferred to the chiplet manufacturer and the system integrator. For example, a unique unclonable identifier might be provided by each interposer. Or, trust might be established in the parts of the supply chain that provide an opportunity to modify an interposer.

Side channel attack of a chiplet.	
Description	An adversary might use micro-probing techniques on individual chiplets in the system to perform passive non-invasive side-channel attacks to extract sensitive information contained in or exchanged between chiplets. Side-channel attacks might employ techniques such as differential power analysis, timing analysis, or measuring electromagnetic emissions. The high amount of interconnection between chiplets in a system and the possible interdependencies between chiplets (for example, a timing reference) might provide a greater sensitivity for these attacks when compared to a monolithic design.
STRIDE category	Information disclosure.
Adversarial model	AM 2.
Security goal	Confidentiality of sensitive information contained in or exchanged between chiplets in a system.
Mitigating action (suggested)	This threat is transferred to the chiplet manufacturer and the system integrator. Employ side-channel attack countermeasures as the application requires them. For example, the physical shielding of sensitive functionality or the use of blinding techniques that obfuscate the operation of some functionality.

Reading data as it is transmitted between chiplets.	
Description	An adversary physically exposes and makes contact with an interface between chiplets to read messages containing protected data.
STRIDE category	Information disclosure.
Adversarial model	AM 4.
Security goal	Encryption of data as it is communicated between chiplets.
Mitigating action (suggested)	The threat is controlled by transferring protected data using interfaces that employ encrypted transmission.

Data extraction through a modified chiplet-to-chiplet interposer.	
Description	An adversary modifies a chiplet-to-chiplet interposer design during manufacture to insert passive signals or active circuits that are accessed when the system is operational through physical probing of the package. These signals or circuits are used to read protected data that is communicated between chiplets.
STRIDE category	Information disclosure.
Adversarial model	AM 6.
Security goal	Encryption of data as it is communicated between chiplets.
Mitigating action (suggested)	The threat is controlled by transferring protected data using interfaces that employ encrypted transmission.

Disruption of system operation through a chiplet-to-chiplet interface.	
Description	An adversary physically exposes and makes contact with an interface between chiplets to disrupt the availability of some functionality of the system.
STRIDE category	Denial of service.
Adversarial model	AM 4.
Security goal	Continued availability of the system, or controlled restriction of availability of the system.
Mitigating action (suggested)	This threat is transferred to the chiplet system integrator. Defensive software behaviors might be employed to limit the impact by, for example, disabling privileged system functionality in response. Or, mechanisms for the detection of interference might initiate a process for recovery using redundant functionality.

Interference with chiplet security lifecycle state.	
Description	Each chiplet in the system might have a security lifecycle state. An adversary might modify the security lifecycle state of an individual chiplet using a physically invasive technique, such that it does not align with the rest of the system and so gain access to sensitive information. For example, a debug interface of a chiplet might be exposed by altering the security lifecycle state to one that is used for provisioning or repair. That debug interface is then used to make privileged accesses to other parts of the system.
STRIDE category	Elevation of privilege.
Adversarial model	AM 4.
Security goal	System level management of chiplet security lifecycle states.
Mitigating action (suggested)	This threat is controlled by employing a process of securely checking the security lifecycle states of all chiplets in the system at a system level.

Injection of privileged control messages through a chiplet-to-chiplet interface.	
Description	An adversary physically exposes and makes contact with an interface between chiplets to inject privileged control messages and disable protection checks or otherwise alter system state.
STRIDE category	Elevation of privilege.
Adversarial model	AM 4.
Security goal	Encryption of privileged communication between chiplets, with integrity and replay protection.
Mitigating action (suggested)	This threat is controlled by restricting the receipt of privileged control messages to interfaces that employ encrypted transmission and are encrypted in a way that guarantees the provenance of the message, and by requiring replay protection that renders ineffective the injection of pre-recorded privileged control messages.

Glossary

ACPI

Advanced Configuration and Power Interface.

AMBA

Advanced Microprocessor Bus Architecture - Arm Interconnect and related standards.

Application PE

A PE, typically A-Profile, that runs the main operating system.

ATB

(AMBA) Advanced Trace Bus, see [19].

ATS

(PCIe) Address Translation Services, see [15].

AXI

Advanced eXtensible Interface, part of the AMBA family of protocols, see [20].

C2C

(AMBA) Chip-to-Chip Interface - for transporting AMBA protocols between chiplets. For more information see [14].

CCA

Arm Confidential Compute Architecture, see [5].

CHI

(AMBA) Coherent Hub Interface, for more information see [3].

CRC

Cyclic Redundancy Check.

CXL

Compute eXpress Link, see [21].

DAP

Debug Access Port, see [7].

DPU

Data Processing Unit.

DVFS

Dynamic Voltage and Frequency Scaling.

DVM

Distributed Virtual Memory.

ETR

Embedded Trace Router, a component of the CoreSight trace infrastructure, see [8].

GIC

Generic Interrupt Controller, see [4].

GICD

GIC Distributor, a component of the GIC, see [4].

GPC

Granule Protection Check (see FEAT_RME in the Arm Architecture [2]).

GPT

Granule Protection Table (see FEAT_RME in the Arm Architecture [2]).

HES

Hardware Enforced Security.

I3C

Improved Inter-Integrated Circuit (Interface), see [13].

IEEE

Institute of Electrical and Electronics Engineers.

ITS

Interrupt Translation Service, a component of the GIC, see [4].

JTAG

Joint Test Action Group.

KGD

Known Good Die.

LPI

Locality-Specific Peripheral Interrupt, an interrupt type specified by the GIC architecture, see [4].

MCM

Multi-Chip Module, see https://en.wikipedia.org/wiki/Multi-chip_module.

MHU

Message Handling Unit.

MMU

Memory Management Unit.

MPAM

(Arm) **M**emory System Resource **P**artitioning **A**nd **M**onitoring, see [22].

MPE

Memory Protection Engine (see FEAT_RME in the Arm Architecture [2]).

MPMM

Max Power Mitigation Mechanism.

MSI

Message Signalled Interrupt.

NRE

Non-Recurrent Engineering (cost).

PA

Physical Address - An address that identifies a location in the physical memory map.

PCIe

Peripheral Component Interconnect Express, see [15].

PCSA

Power Control System Architecture [23].

PE

Processing Element - A processing element that implements the Arm architecture, see [2].

PPU

Power Policy Unit, see [11], for further details of use of the PPU see the PCSA [23].

RAS

Reliability, Availability and Serviceability.

RE

Recurring Engineering (Cost).

RME

Realm Management Extension (see FEAT_RME in the Arm Architecture [2]).

RME-CDA

RME-Coherent Device Assignment.

RME-DA

RME-Device Assignment.

RoT

Root of Trust.

SCMI

System Control and Management Interface, see [17].

SiP

System-in-Package.

SMMU

System MMU, an MMU for system components, see [9].

SoC

System-on-Chip.

SPI

Shared Peripheral Interrupt, an interrupt type specified by the GIC architecture, see [4].

SWO

Serial Wire Output.

TAP

Test Access Port, part of the JTAG / *IEEE Std 1149.1™-2001* standard, see [24].

TCK

Test Clock, a standard TAP interface signal.

TDI

Test Data In, a standard TAP interface signal.

TDO

Test Data Out, a standard TAP interface signal.

TDP

Thermal Design Power.

TMS

Test Mode Select, a standard JTAG interface signal.

TPIU

Trace Port Interface Unit, see [7].

UCIe

Universal Chiplet Interface Express, for more information see [12].

UEFI

Unified Extensible Firmware Interface.