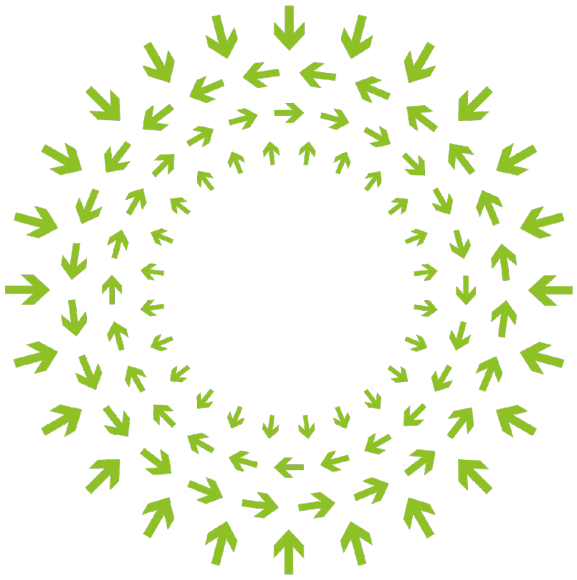

OPEN

Compute Project

Foundation Chiplet System Architecture

1.0.0

Effective February 2026

Author: Arm Ltd.

Contents

Foundation Chiplet System Architecture

Version History . . . vi
License . . . vii
Open Web Foundation (OWF) CLA . . . vii
Trademarks . . . viii
Compliance with OCP Tenets . . . ix
Openness . . . ix
Efficiency . . . ix
Impact . . . ix
Scale . . . ix
Sustainability . . . ix
Rules-based writing . . . x
Content item identifiers . . . x
Content item rendering . . . x
Content item classes . . . x
Additional reading . . . xi

1 Introduction . . . 1
1.1 Layered Architecture . . . 2
1.2 Purpose . . . 3
1.2.1 Document Structure . . . 3
1.3 Using this specification . . . 4
1.3.1 System Designer Considerations . . . 4
1.3.2 Standalone Chiplet Designer Considerations . . . 4
1.3.3 Chiplet types and classification . . . 5
1.3.4 Interface types and classification . . . 5
1.4 Basic Definitions . . . 6

2 Rule Compliance Levels . . . 7
2.1 Overview . . . 8
2.1.1 Level 0 . . . 8
2.1.2 Level 1 . . . 8
2.1.3 Full Compliance . . . 8
2.2 Rule Identifiers by Level . . . 9

3 System Types . . . 15
3.1 Compute and Hub System . . . 16
3.1.1 System Requirements . . . 18
3.2 Compute Tile System . . . 19
3.2.1 System Requirements . . . 21

4 Chiplet Types . . . 22
4.1 Compute 1 Chiplet . . . 23
4.1.1 Chiplet Requirements . . . 24
4.2 Hub Chiplet . . . 27

4.2.1 Chiplet Requirements . . . 28
4.3 Compute 2 Chiplet . . . 32
4.3.1 Chiplet Requirements . . . 33
4.4 Fully Coherent Expansion Chiplet . . . 38
4.4.1 Chiplet Requirements . . . 39
4.5 Fully Coherent Expansion (Remote Translation) Chiplet . . . 43
4.5.1 Chiplet Requirements . . . 44
4.6 I/O Coherent Expansion (Translated) Chiplet . . . 47
4.6.1 Chiplet Requirements . . . 48
4.7 I/O Coherent Expansion (Untranslated) Chiplet . . . 51
4.7.1 Chiplet Requirements . . . 52
4.8 I/O Coherent Expansion (Remote Translation) Chiplet . . . 55
4.8.1 Chiplet Requirements . . . 56
4.9 I/O Chiplet . . . 59
4.9.1 Chiplet Requirements . . . 60
4.10 I/O Controller Chiplet . . . 62
4.10.1 Chiplet Requirements . . . 63

5 Chiplet Interfaces . . . 66
5.1 Chiplet Activity States and Interface Availability . . . 67
5.2 Hub Chiplet to Hub Chiplet Interface Requirements . . . 69
5.2.1 Boot Code Image Load Interface . . . 69
5.2.2 Coherent Memory Traffic Interface . . . 69
5.2.3 Interrupt Handling . . . 70
5.2.4 Security . . . 70
5.2.5 System Control Communication . . . 70
5.2.6 Debug . . . 71
5.3 Hub Chiplet to Compute Chiplet Interface Requirements . . . 72
5.3.1 Boot Code Image Load Interface . . . 72
5.3.2 Coherent Memory Traffic Interface . . . 72
5.3.3 Interrupt Handling . . . 73
5.3.4 Security . . . 73
5.3.5 System Control Communication . . . 73
5.3.6 Debug . . . 74
5.4 Compute 1 Chiplet to Compute 1 Chiplet Interface Requirements . . . 75
5.4.1 Boot Code Image Load Interface . . . 75
5.4.2 Coherent Memory Traffic Interface . . . 75
5.4.3 Interrupt Handling . . . 76
5.4.4 Security . . . 76
5.4.5 System Control Communication . . . 76
5.4.6 Debug . . . 77
5.5 Compute 2 Chiplet to Compute 2 Chiplet Interface Requirements . . . 78
5.5.1 Boot Code Image Load Interface . . . 78
5.5.2 Coherent Memory Traffic Interface . . . 78
5.5.3 Interrupt Handling . . . 79
5.5.4 Security . . . 79
5.5.5 System Control Communication . . . 79
5.5.6 Debug . . . 80
5.6 Fully Coherent Expansion Chiplet Interface Requirements . . . 81
5.6.1 Boot Code Image Load Interface . . . 81
5.6.2 Coherent Memory Traffic Interface . . . 81
5.6.3 Interrupt Handling . . . 82
5.6.4 Device Configuration Interface . . . 82
5.6.5 Security . . . 82
5.6.6 System Control Communication . . . 83

5.6.7 Debug . . . 83
5.7 Fully Coherent Expansion (Remote Translation) Chiplet Interface Requirements 84
5.7.1 Coherent Memory Traffic Interface . . . 84
5.7.2 I/O Coherent Memory Traffic Interface . . . 84
5.7.3 Address Translation Interface . . . 85
5.7.4 Interrupt Handling . . . 85
5.7.5 Device Configuration Interface . . . 85
5.7.6 Security . . . 86
5.7.7 System Control Communication . . . 86
5.7.8 Debug . . . 87
5.8 I/O Coherent Expansion (Translated) Chiplet Interface Requirements . . . 88
5.8.1 Boot Code Image Load Interface . . . 88
5.8.2 I/O Coherent Memory Traffic Interface . . . 88
5.8.3 Non-Coherent Memory Traffic Interface . . . 89
5.8.4 Interrupt Handling . . . 89
5.8.5 Device Configuration Interface . . . 89
5.8.6 Security . . . 90
5.8.7 System Control Communication . . . 90
5.8.8 Debug . . . 90
5.9 I/O Coherent Expansion (Untranslated) Chiplet Interface Requirements . . . 91
5.9.1 Boot Code Image Load Interface . . . 91
5.9.2 Untranslated I/O Coherent Memory Traffic Interface . . . 91
5.9.3 Interrupt Handling . . . 92
5.9.4 Device Configuration Interface . . . 92
5.9.5 Security . . . 92
5.9.6 Debug . . . 93
5.10 I/O Coherent Expansion (Remote Translation) Chiplet Interface Requirements 94
5.10.1 I/O Coherent Memory Traffic Interface . . . 94
5.10.2 Address Translation Interface . . . 94
5.10.3 Non-Coherent Memory Traffic Interface . . . 95
5.10.4 Interrupt Handling . . . 95
5.10.5 Device Configuration Interface . . . 95
5.10.6 Security . . . 96
5.10.7 System Control Communication . . . 96
5.10.8 Debug . . . 96
5.11 I/O Chiplet Interface Requirements . . . 98
5.11.1 Boot Code Image Load Interface . . . 98
5.11.2 Untranslated I/O Coherent Memory Traffic Interface . . . 98
5.11.3 Non-Coherent Memory Traffic Interface . . . 99
5.11.4 Interrupt Handling . . . 99
5.11.5 Device Configuration Interface . . . 99
5.11.6 Security . . . 100
5.11.7 Debug . . . 100
5.12 I/O Controller Chiplet Interface Requirements . . . 101
5.12.1 Boot Code Image Load Interface . . . 101
5.12.2 Non-Coherent Memory Traffic Interface . . . 101
5.12.3 Interrupt Handling . . . 102
5.12.4 Device Configuration Interface . . . 102
5.12.5 Security . . . 102
5.12.6 Debug . . . 103

6 Common Chiplet Components . . . 104
6.1 System Control Framework . . . 105
6.1.1 System Controller . . . 105
6.1.2 System Control Agent . . . 106

6.2 Trusted Security Agent . . . 107

7 Chiplet Interface Implementation . . . 109
7.1 Hardware Fault Requirements for Chiplet Interfaces . . . 110
7.2 Interface Protocols . . . 111
7.2.1 Boot Code Image Load Interface . . . 111
7.2.2 Coherent Memory Traffic Interface . . . 111
7.2.3 Untranslated I/O Coherent Memory Traffic Interface . . . 111
7.2.4 I/O Coherent Memory Traffic Interface . . . 111
7.2.5 Non-Coherent Memory Traffic Interface . . . 111
7.2.6 Device Configuration Interface . . . 112
7.2.7 Address Translation Interface . . . 112
7.2.8 Interrupt Handling . . . 113
7.2.9 Trusted Security Agent Communication Interface . . . 113
7.2.10 System Control Interface . . . 113
7.2.11 Debug . . . 113
7.3 Interface Transports . . . 114
7.3.1 Boot Code Image Load Interface . . . 114
7.3.2 Coherent Memory Traffic Interface . . . 114
7.3.3 Untranslated I/O Coherent Memory Traffic Interface . . . 114
7.3.4 I/O Coherent Memory Traffic Interface . . . 114
7.3.5 Non-Coherent Memory Traffic Interface . . . 115
7.3.6 Device Configuration Interface . . . 115
7.3.7 Address Translation Interface . . . 116
7.3.8 Interrupt Handling . . . 116
7.3.9 Trusted Security Agent Communication Interface . . . 117
7.3.10 System Control Interface . . . 117
7.3.11 Debug . . . 117

A OCP Checklist . . . 119

Glossary . . . 120

Version History

Date	Version #	Author	Description
19 Nov 25	1.0.0 Draft 0	Arm Ltd.	First release.
22 Jan 26	1.0.0 Draft 1	Arm Ltd.	Updated System Control Description. Relaxed Rules on UCIe. Relaxed Rules on CHI-C2C.
27 Jan 26	1.0.0 Draft 2	Arm Ltd.	Added trademarks. Corrected document style.
10 Feb 26	1.0.0	Arm Ltd.	Release of 1.0.0.

License

Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in **Modified Open Web Foundation Agreement 0.9 (OWFa 0.9)**. (As of October 16, 2024) ("Contribution License") by:

Arm Ltd.

Usage of this Specification is governed by the terms and conditions set forth in **Modified OWFa 0.9 Final Specification Agreement (FSA)** (As of October 16, 2024) **("Specification License").**

You can review the applicable Specification License(s) referenced above by the contributors to this Specification on the OCP website at https://www.opencompute.org/contributions/templates-agreements.

For actual executed copies of either agreement, please contact OCP directly.

Notes:

The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Trademarks

AMBA® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

CXL® is a registered trademark of Compute Express Link Consortium, Inc.

HDMI® is a registered trademark of HDMI Licensing Administrator, Inc.

I3C® is a registered trademark of MIPI Alliance, Inc.

MIPI CSI-2® is a registered trademark of MIPI Alliance, Inc.

PCIe® and PCI Express® are registered trademarks of PCI-SIG.

RISC-V® is a registered trademark of RISC-V International.

UCIe™ is a trademark of UCIe (Universal Chiplet Interconnect Express).

All other brands or product names are the property of their respective holders.

Compliance with OCP Tenets

This section describes how this specification complies with the OCP tenets.

A full explanation of the OCP core tenets can be seen here.

Openness

The FCSA is licensed under the Open Web Foundation Agreement (OWFa) version 0.9. Feedback and contributions are encouraged from the OCP community, and the specification is reviewed and developed under the supervision of the public Foundation Chiplet System Architecture workstream, operating under the Open Chiplet Economy Sub-Project.

Efficiency

The FCSA enhances efficiency by encouraging and facilitating the re-use of silicon dies in multiple systems and packages, bringing economies of scale to SoC/SiP development and manufacturing processes, and reducing the cost and time to market for custom semiconductor solutions.

Impact

The FCSA provides a new baseline for the chiplet market, empowering the industry to build interoperable chiplets across domains. The FCSA defines common system topologies, standard chiplet types, and functional and interface requirements between chiplets, to provide a common foundation to maximize interoperability and reduce fragmentation across the industry.

Scale

Multi-die systems built using chiplets provide a route for common dies to be used in different configurations and topologies to scale up performance while retaining an acceptable yield. The FCSA supports features required in large scale deployments including error reporting, manageability, and advance security technologies.

Sustainability

The FCSA embraces sustainability by reducing development and manufacturing costs for advanced custom semiconductor systems.

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.

It is strongly recommended 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

[1] *Open Compute Project*. See https://www.opencompute.org/

[2] *Open Chiplet Econmomy OCP Subproject*. See https://www.opencompute.org/projects/open-chiplet-economy

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

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

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

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

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

[8] *RISC-V Platform Management Interface Specification (RPMI)*. RISC-V International.

[9] *Bunch of Wires (BoW) PHY Specification*. Open Compute Project (OCP).

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

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

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

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

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 Foundation Chiplet System Architecture (FCSA) is applicable for a monolithic system design partitioned across chiplets, where chiplet interfaces replace on-die connections. For partitioned systems, the FCSA 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, there is limited additional value that can be added by the FCSA. Both approaches are equally important for building chiplet-based systems.

This Foundation Chiplet System Architecture document specifies a hardware system architecture for a system that is distributed over multiple chiplets. A *system* is defined as the hardware that supports a single system software image such as an operating system.

The FCSA 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 Layered Architecture

This architecture specification is part of a layered system architecture framework as shown in Figure 1.1. There is a contribution hierarchy in which higher layers add to the contribution of lower layers to form a more complete specification, targeted at a particular design or product profile.

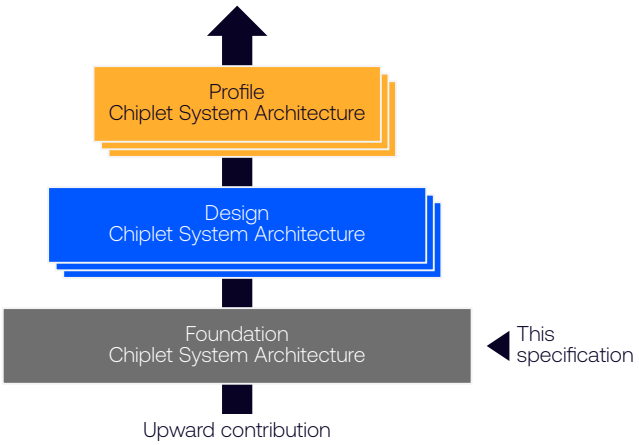

Figure 1.1: Layered Architecture

This specification is the foundation specification. These are the layers:

Foundation

This specification.

The Foundation Chiplet System Architecture (FCSA) provides alignment to the system architecture by specifying basic requirements and common concepts that are not specific to a design or product profile.

The FCSA is a contribution to the Open Compute Project (OCP) [1]. Further information, and details of how to make a contribution to the FCSA, is available at the Open Chiplet Economy OCP subproject [2].

Design

A design specification builds upon the FCSA with the addition of requirements that align the system architecture with a particular system design. For a system built using chiplets, this architecture might be specific to a particular ISA.

Profile

A profile specification builds upon a design specification to target the system architecture at a certain product profile. This could be a market segment, a specialisation for an environment in which the product is to be deployed, or a narrowly focussed functional extension.

Higher-level specifications incorporate the contributions of lower layers and add further requirements. Because this document is rules-based, each higher-level specification's rule set is a superset of the rule sets defined by the lower-level specifications it extends.

1.2 Purpose

This specification is intended to act as a foundation for a 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 FCSA provides a consistent implementation. This enables chiplet compatibility without limiting differentiation.

1.2.1 Document Structure

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

Figure 1.2 shows the relationship between the components specified in the FCSA.

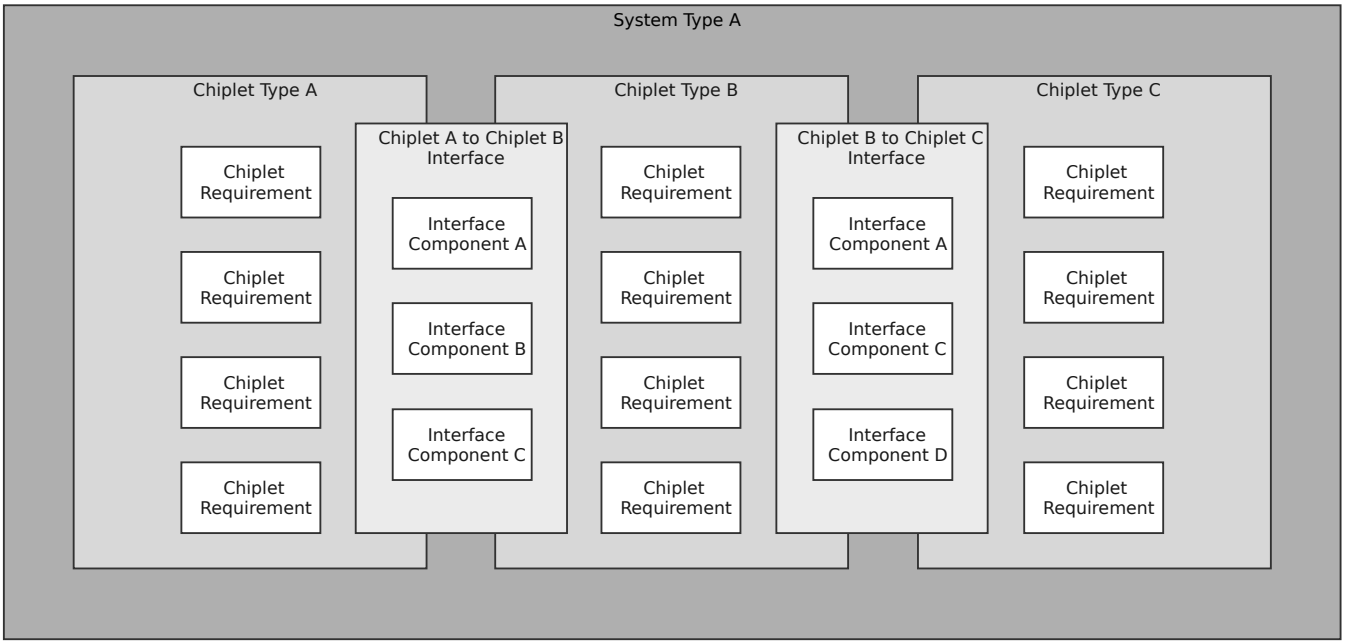

Figure 1.2: FCSA Component Structure

A system is distributed across multiple chiplets, each chiplet is of a defined type. The FCSA 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 FCSA specifies protocols and transport mechanisms where appropriate industry standards exist, otherwise the FCSA maps the interface component as TO BE DEFINED.

1.3 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 FCSA 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.

1.3.1 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 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 Request for Quote (RFQ) for a third party chiplet designer.

1.3.2 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 is 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 FCSA.

1.3.3 Chiplet types and classification

A FCSA 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.

Supporting multiple chiplet types can increase a chiplet's flexibility, allowing the chiplet to be used in more than one system. However, this approach may be suboptimal in a given deployment, as functionality included to support chiplet types other than the one in use remains unused. This unused functionality represents a potential waste of silicon area and power.

1.3.4 Interface types and classification

A FCSA chiplet interface type is defined by a set of requirements that an implementation must meet. These requirements are determined by the chiplet types that are being connected and they ensure that the defined chiplet types are able to interact and to create a functional system.

An interface may be able to interact with multiple chiplet types, provided that both the interface and the chiplets support the required functions. In such cases, a single interface could be classified as more than one type. This may require the interface to be configured - either during manufacturing or at runtime - to support connection to a specific chiplet type.

Supporting multiple interface types increases flexibility in how an interface - and therefore the chiplet - can be used. However, this can be suboptimal in a given deployment, as functionality included to support unused interface types may waste silicon area and power.

1.4 Basic Definitions

The following terms or concepts are used throughout the FCSA.

System

The hardware that supports a single system software image of an operating system or hypervisor.

Application PE

A Processing Element (PE) that runs the operating system.

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 FCSA 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 referred to by the FCSA as “System Addressable Memory”. The FCSA assumes that all Application PEs in the system share the same operating system or hypervisor.

System Main Memory

Memory that is both:

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

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.

Cacheable system main memory

Locations in *System Main Memory* that are allowed to be allocated in a cache.

Non-cacheable system main memory

Locations in *System Main Memory* that are not allowed to be allocated in a cache.

Boot Code Image

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

2
Rule Compliance Levels

I_NDLSV This section describes the levels of compliance within the FCSA and how they can be applied.

2.1 Overview

The Foundation Chiplet System Architecture (FCSA) 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$_{YYYYY}$ This is an example of a FCSA 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 FCSA.

2.1.1 Level 0

This level means the chiplet meets the basic functional and interface rules for the chiplet or system type.

A list of the level 0 rule identifiers that appear in the FCSA 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 additional requirements appropriate for the chiplet type or system.

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

2.1.3 Full Compliance

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

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

2.2 Rule Identifiers by Level

Table 2.1: Level 0 Rule Identifiers in the FCSA

3 System Types							
3.1 Compute and Hub System							
LYHKS	TFJSM	LCJNF	CJZHL	ZCFRY	PPLNQ	MDPNF	FZVCL
DXYWQ	TZJGV	ZKZHH					
3.2 Compute Tile System							
TKPTF	MVWBD	JHWVM	DSKBW	ZFTDV	BFSJV	ZDVGZ	CCHPT
TTPHX	WTRKM						
4 Chiplet Types							
4.1 Compute 1 Chiplet							
JWRCG	BPRJK	RCYDP	XTDHG	PTZSC	NCHRQ	KNKSV	NRJZH
NGZDK	ZHHPH	XGXZR	MCLJS	CQDKD	GHRFJ	ZYTGQ	TGSXZ
MJNJG	GVVMS	ZDSHM	DZFZV	RNSKY	PGGRB	ZBRCQ	XSFHF
SFPHM	MDGWP	MRRGT					
4.2 Hub Chiplet							
MZLJX	CNQZH	LCQGC	SVWVP	HBYND	MKYNH	WMTVP	MBQZT
MLQLD	ZTCKN	PCJMB	FXFNL	JDYJY	PMKFN	YHQDR	XDGQV
RXLZG	MTBPW	VJFWY	KZFYG	RGMKW	GNJLK	GSSWV	FTXZM
ZPLGW	VNGDP	RFPQN	NJQDV	NXDVW	CKXYX	ZFWWC	PYRRZ
BTGJS	ZCPHB	LDSVP	GYSJB	SXNDJ	YJBND	KYYNQ	DZGQK
4.3 Compute 2 Chiplet							
VKGJK	SJNGN	NDHDT	KNHMM	CGKQW	MDFWD	WLHVZ	JZSRC
DVNHD	QGGWQ	LGXLN	XNYDJ	PHHQM	ZVZZN	VYJFL	ZCLCN
DMSQX	XCVGN	ZXWKH	DZDHQ	PBYQY	CLYNB	GPSSB	TXVCM
THPPG	NYZVM	YNMLX	SGZXK	VVDFD	NLGDR	WVMSK	PGTMQ
YQMHG	FDRRV	CNDVJ	FWCJG	KGFNL	GNZNV	ZDBLR	XDDFR
XYVWT	LPTTQ						
4.4 Fully Coherent Expansion Chiplet							
SHXTT	HXGDG	GHKCL	MKCTS	HMZJQ	YDWVV	NBXWR	FMBVG

JNCKH	VCPQR	FJDNP	SPTFW	FQPML	QDMPX	LYVBX	LVJXD
XVWQF	WHJVN	QQTXS	RDGJV	PBNCH	FDDHQ	FXKVK	ZRPHF
QZSYG	PWHNJ	RNNVV	ZNXXR	RJMNL	XLWKP	FFWWY	BMPMM
ZDSMN	VJGGP						

4.5 Fully Coherent Expansion (Remote Translation) Chiplet

KGJQZ	VZRNQ	ZKKWN	VGSNV	KSDBW	MHJBP	GGJVW	WDGDV
KLQNT	WNHGS	VHJYG	SVMSC	FFDXH	XMDFB	LMWRD	DDBSL
PJVLK	RVBSS	CWKWQ	NYNCY	MBSCP	XVZTS	JSDXW	XBJSZ

4.6 I/O Coherent Expansion (Translated) Chiplet

YTTPM	MPKFL	LGNWX	RWJXM	NJQMS	FHVQW	XPQFB	DFJJB
LRYYF	NKBDT	GPNPF	JKNBM	YLRSX	TXQKB	PGQRS	XCBFQ
HSWKR	BGYTR	XHLRJ	RDSRT	ZHDSR	LPFHQ	SXYNL	SDNYR
TWRHN	ZHTQD	PHQVC	LCWDK	JTTCQ	ZKGSN	DDQVP	

4.7 I/O Coherent Expansion (Untranslated) Chiplet

YWWFY	GNBSW	PZFMT	WFLXB	JHQPD	BNCHJ	DGKSX	HKCGS
YHDSL	DPLFM	NDGMQ	RGMBN	NPHFD	KFPMT	PQGLG	RBSQV
KMTCS	JMJYS	HHJCC	MGPTM	PJLCZ	WSWRW		

4.8 I/O Coherent Expansion (Remote Translation) Chiplet

MCTLV	JFVWW	LYDWT	LVZJJ	YKSJV	DFKTT	XKHNG	RXVFP
SJJRK	VGYHF	GXVLS	MGPMV	MJDDS	VLFCJ	GNNND	JWZKZ
SSKPN	XCBFD	RHJMG	PWMCT	VVWXF	MTJGB	WXTJK	DZVCK

4.9 I/O Chiplet

FKNWS	WWGKL	HCYCZ	CFTXC	FHZFP	ZQPNC	JHMFZ	KWXTX
CPTMG	ZMHCY	RTPGY	YRKBX	JSKVP	RRCYS	MGCKS	LCHML
XJFYN	WPDRB	BNDMC	YFNFR	BXGXT			

4.10 I/O Controller Chiplet

FFNKP	LTWZF	JWNXJ	RDQYR	DSBQD	WBVMW	XWTXV	BNQNG
JRSRT	RZGDN	QWLMP	LZRDM	MWPHD	DWLLK	QMDXK	MDBRR
KZQVC	PDQHN	GRYYV	JDSWQ	NHPSD	VVBCX		

5 Chiplet Interfaces

5.2 Hub Chiplet to Hub Chiplet Interface Requirements

SYDRX LQZMW DFVLC CQVLD DHBHJ TPSJG DKMDM RVKPZ
NJPCS PMLBM BMRTC

5.3 Hub Chiplet to Compute Chiplet Interface Requirements

FVFQQ GHLHT CLWWD GMZTH NNQCD FXRMW SZZGL YGWKV
BHDVN HJZYS HZCKZ

5.4 Compute 1 Chiplet to Compute 1 Chiplet Interface Requirements

LPBNG KYGBK SDKFD JSDGJ KFDDX YYBCT LPQPB SSDSP
FZBGN SCNVZ MFVBR

5.5 Compute 2 Chiplet to Compute 2 Chiplet Interface Requirements

RBNSS CZLQD NLNCD JRLRN TXHMX LLTTW GZMKP RTCSK
PDJZG NLZFW QLHKH

5.6 Fully Coherent Expansion Chiplet Interface Requirements

VLQPR RBMCH WGSPV PFZGF GJBYM JZNWW JZKYZ RVDHX
ZCQQH GHGXC GQTVW LJRBQ FZVDD

5.7 Fully Coherent Expansion (Remote Translation) Chiplet Interface Requirements

CMRXD TGCLP QSHDG SXWZM VBVKL LHSRL LWJBR HYTDS
GLVZD LXTVM GRXQH CSFZB GGTMJ QBXFV GQNNR XXTQB
PCWPV

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

HCYKF HXNJH RGQKJ YHSGY JSFTT BMJZP RNYKY XRDCX
ZGDGY JJKSQ BHRHJ NTZHB SQLJT CYHBV

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

CBWZD SZXHJ QBQMD DTDLR CDBPS RYVZX QTJWG BSQPC
MCCDT PVDQS PDJPV JJPDD

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	QXZDW	VPNZQ
SDPSD							

5.11 I/O Chiplet Interface Requirements

LHRDK	WLLKW	XKHXM	DVBZV	RSJDZ	DFQYV	YLRCM	HJDFT
RVTJM	KBGTC	HXWGL	HHJRZ	TDQTB			

5.12 I/O Controller Chiplet Interface Requirements

DDDSW	YZHWP	YJGHH	LSJMB	PCWSQ	ZQVFY	YRHHX	HLXNC
LKCWK	XZPFB	RNDMS					

6 Common Chiplet Components

6.1 System Control Framework

TMGXQ	NHBRK	CSHRH	NQZFH	FLDNS

6.2 Trusted Security Agent

FFKWC	MWLGW	RCPHY	JMCGT

Table 2.2: Level 1 Rule Identifiers in the FCSA

5 Chiplet Interfaces

5.2 Hub Chiplet to Hub Chiplet Interface Requirements

ZPXZW	KMPRP

5.3 Hub Chiplet to Compute Chiplet Interface Requirements

NQBSN	YWSTJ

5.4 Compute 1 Chiplet to Compute 1 Chiplet Interface Requirements

SZTCH	RSGWB

5.5 Compute 2 Chiplet to Compute 2 Chiplet Interface Requirements

JRSXS	RLQYC

5.6 Fully Coherent Expansion Chiplet Interface Requirements

PRKPY	WPZNH

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

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

5.11 I/O Chiplet Interface Requirements
WGMMT GBYYK

5.12 I/O Controller Chiplet Interface Requirements
RDVNB JHJVP

6 Common Chiplet Components

6.1 System Control Framework
WRJNW

6.2 Trusted Security Agent
LDZYP

7 Chiplet Interface Implementation

7.2 Interface Protocols
JCLGK ZKQNK WVPQS TZRTL

Table 2.3: Level FULL Rule Identifiers in the FCSA

6 Common Chiplet Components
6.1 System Control Framework
BZMMM GGJPQ
6.2 Trusted Security Agent
HRPTP NPXGM VCTWQ PZFPH
7 Chiplet Interface Implementation
7.2 Interface Protocols
KJVCD DPWJP TQRTF DDXMP VLQZG RJNQG CWKYG TDLVS PXBTX LXCQZ

7.3 Interface Transports

GJBYZ	ZZXCL	QDDHW	NCZJN	BCQKB	YNCSL	JDCPH	CJBHK
CTJXM	ZRGXC	HSZXD	BGHPZ	LQPGR	XYZYJ	GTDWN	HKNHZ

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 FCSA 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 PCIe 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.

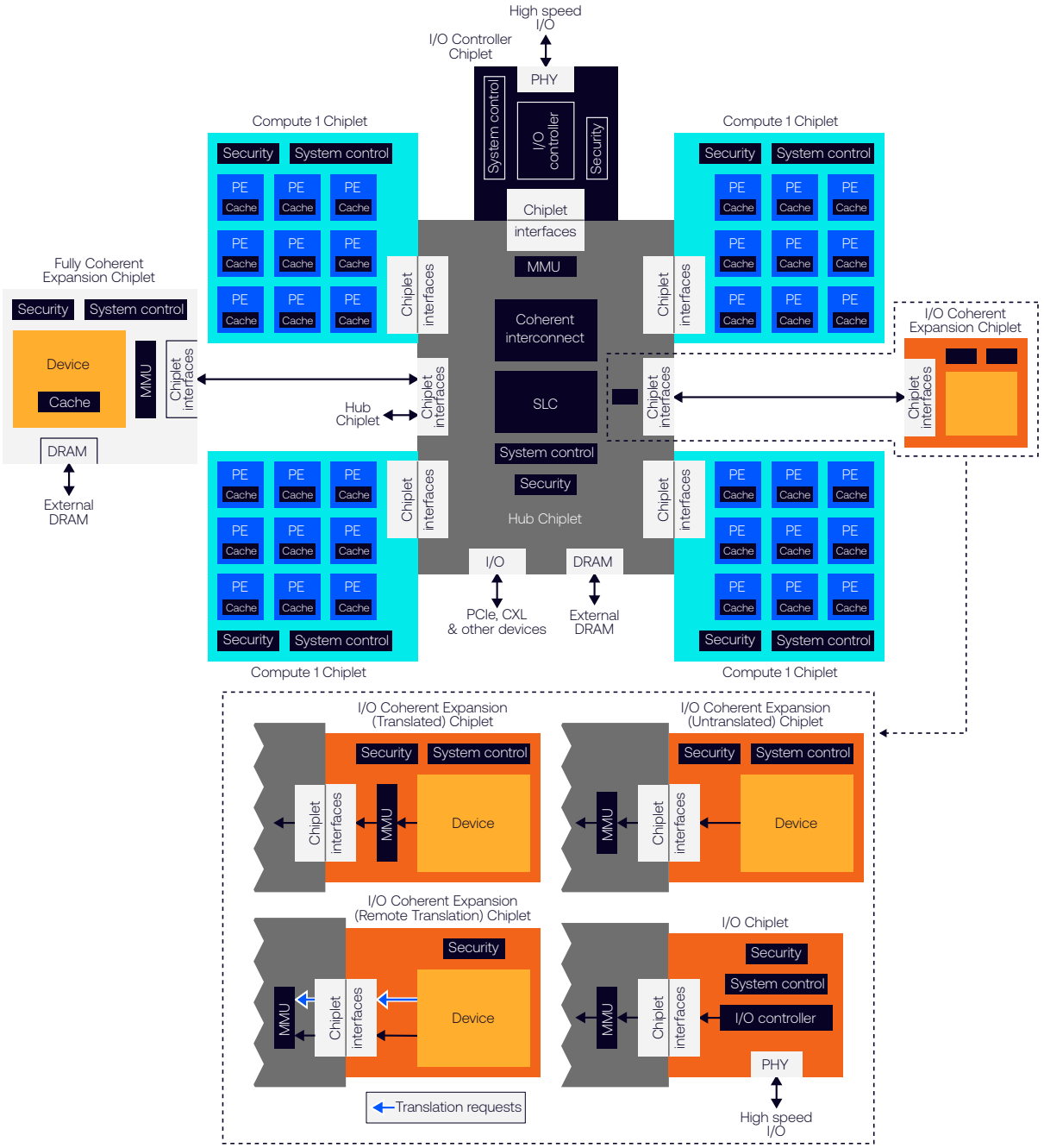

Figure 3.1: Example Compute & Hub System Chiplet Configuration

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

R_{TZJGV} All Application PEs in a Compute & Hub System are coherent with regard to a single commonly shared region of system memory.
(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)

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 PCIe 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.

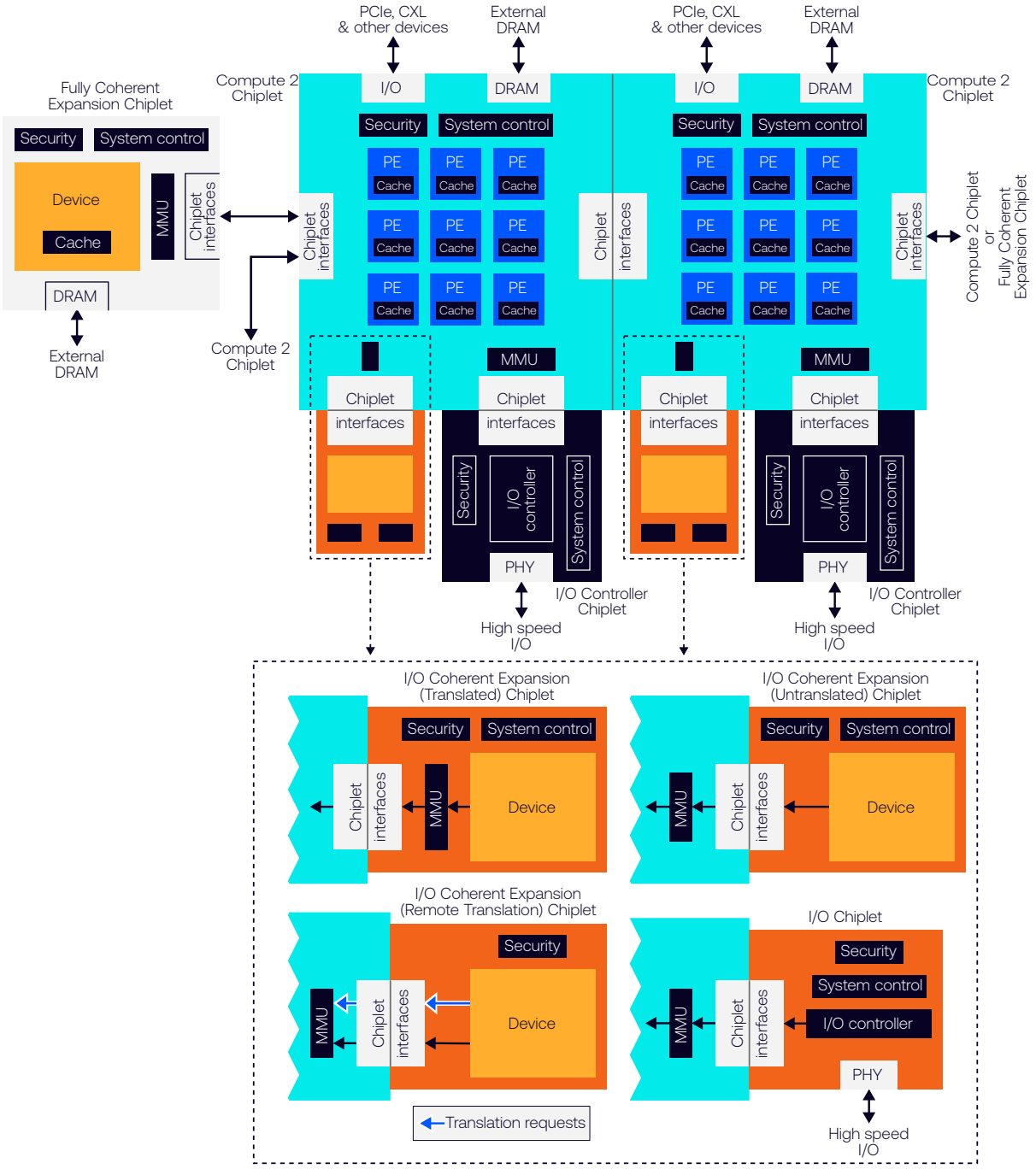

Figure 3.2: Example Compute Tile System Chiplet Configuration

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} A Compute Tile System contains one or more Compute 2 Chiplets.

(LEVEL-0)

R_{MVWBD} A Compute Tile System contains zero or more Fully Coherent Expansion Chiplets.

(LEVEL-0)

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

(LEVEL-0)

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

(LEVEL-0)

R_{ZFTDV} A 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} A Compute Tile System contains zero or more I/O Chiplets.

(LEVEL-0)

R_{CCHPT} A Compute Tile System contains zero or more I/O Controller Chiplets.

(LEVEL-0)

3.2.1.2 Application PE

R_{TTPHX} All Application PEs in a Compute Tile System are coherent with regard to a single commonly shared region of system memory.

(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)

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.

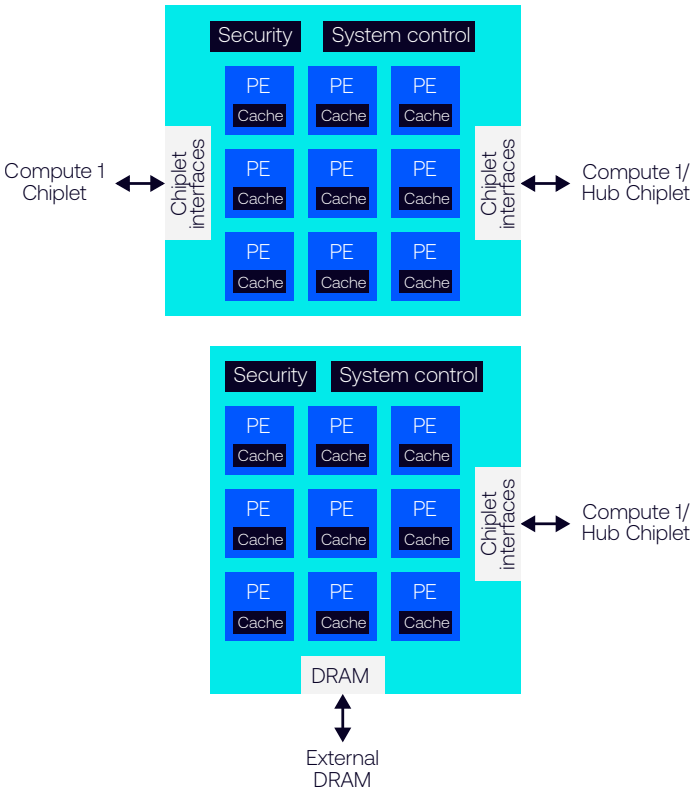

Figure 4.1: Compute 1 Chiplet Examples

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 Chiplet Requirements

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

4.1.1.1 Chiplet-to-Chiplet Interfaces

R_{JWRCG} A Compute 1 Chiplet is interfaced with one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 1 Chiplet.

(LEVEL-0)

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 exactly one Hub 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_{ZHHPH} A Compute 1 Chiplet might contain 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_{MCLJS} Any system addressable memory in the system is available through a Compute 1 Chiplet to 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.

R_ZYTGQ PEs in a Compute 1 Chiplet access system memory through an MMU. That MMU is contained in the chiplet.

(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.

R_TGSXZ A Compute 1 Chiplet contains an interrupt controller component.

(LEVEL-0)

I_CTFCS Interrupts are received at an interrupt controller 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 interrupt controller component 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 an interrupt controller component.

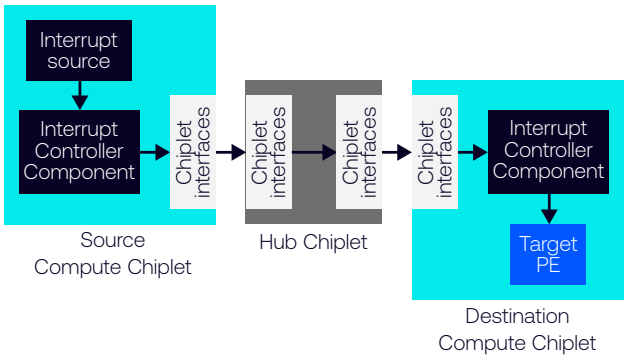

Figure 4.2: Example interrupt communication between two compute chiplets

R_MJNJG The Compute 1 Chiplet interrupt controller component receives interrupts from within the same chiplet.

(LEVEL-0)

R_GVVMS The Compute 1 Chiplet interrupt controller component receives interrupts for the PEs it contains, forwarded from other chiplets.

(LEVEL-0)

R_{ZDSHM} Where the target PE of an interrupt is in another chiplet, the Compute 1 Chiplet interrupt controller component forwards the interrupt to the interrupt controller component in the chiplet that contains the target PE.

(LEVEL-0)

R_{DZFZV} Where the target PE of an interrupt is in the same chiplet, the Compute 1 Chiplet interrupt controller component forwards interrupts to the destination PE.

(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)

R_{ZBRCQ} A Compute 1 Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.1.1.10 Security

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 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)

4.1.1.12 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.

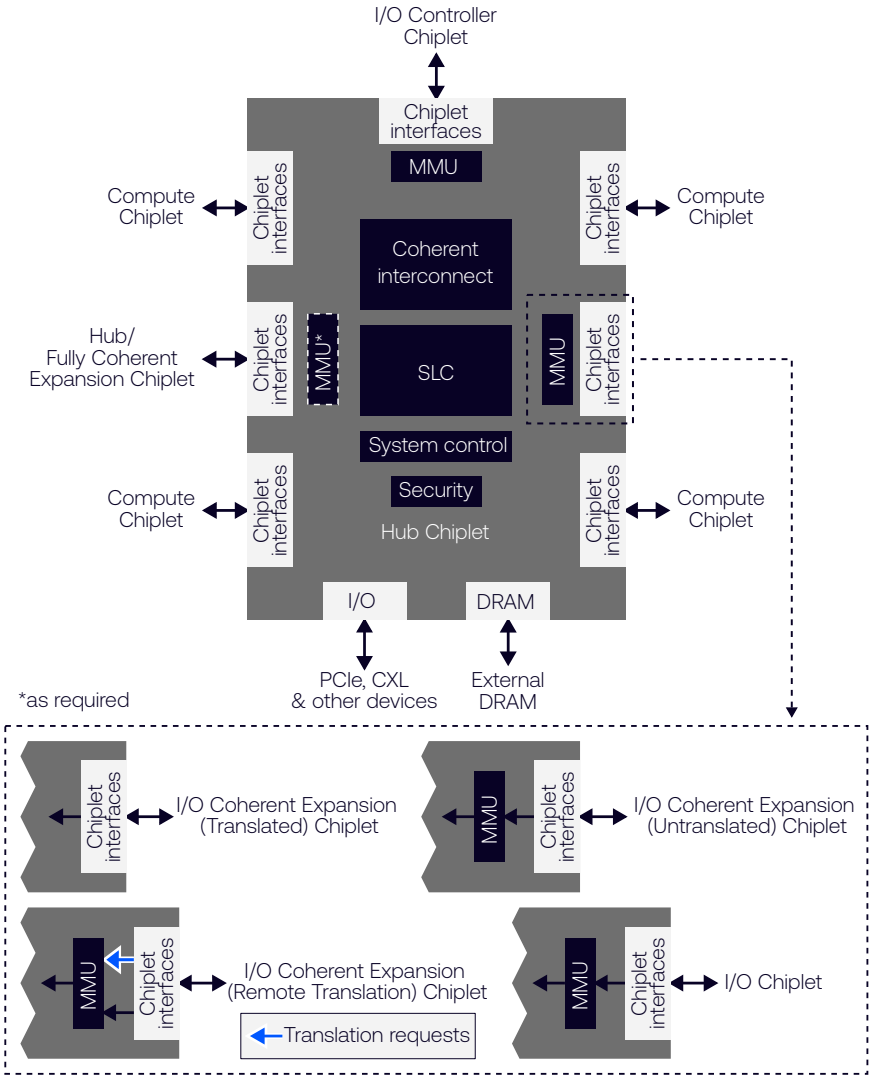

Figure 4.3: Hub Chiplet Example

The Hub 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 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_{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$_{XDGQV}$ A Hub Chiplet contains 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 receive 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.

R$_{RGMKW}$ I/O coherent or non-coherent agents in a Hub Chiplet access system memory through an MMU. That MMU is contained in the chiplet.

(LEVEL-0)

R$_{GNJLK}$ Fully coherent agents in a Hub Chiplet access system memory through an MMU. That MMU is contained in the 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.

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 that does not have an interrupt controller component, with the destination being a PE in another chiplet.

R$_{GSSWV}$ A Hub Chiplet contains an interrupt controller component.

(LEVEL-0)

I$_{PPCWK}$ Interrupts are received at an interrupt controller component in the Hub Chiplet and then forwarded to the interrupt controller component in the chiplet containing the target PE.

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

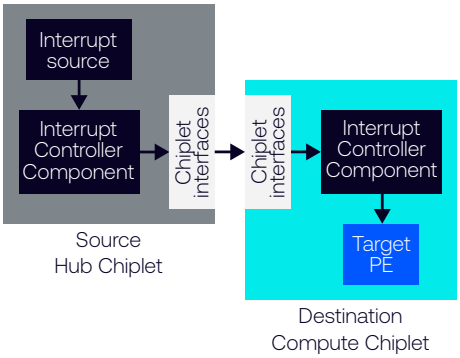

Figure 4.4: Example interrupt communication between a hub and a compute chiplet

R_{FTXZM} The Hub Chiplet interrupt controller component receives interrupts from within the same chiplet.

(LEVEL-0)

R_{ZPLGW} The Hub Chiplet interrupt controller component receives interrupts from other chiplets in the system that do not contain an interrupt controller component.

(LEVEL-0)

R_{VNGDP} The Hub Chiplet interrupt controller component forwards interrupts to the interrupt controller component in the chiplet of the target PE.

(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 images 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)

R_{BTGJS} A Hub Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.2.1.11 Security

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$_{LDSVP}$ 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 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)

4.2.1.13 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.

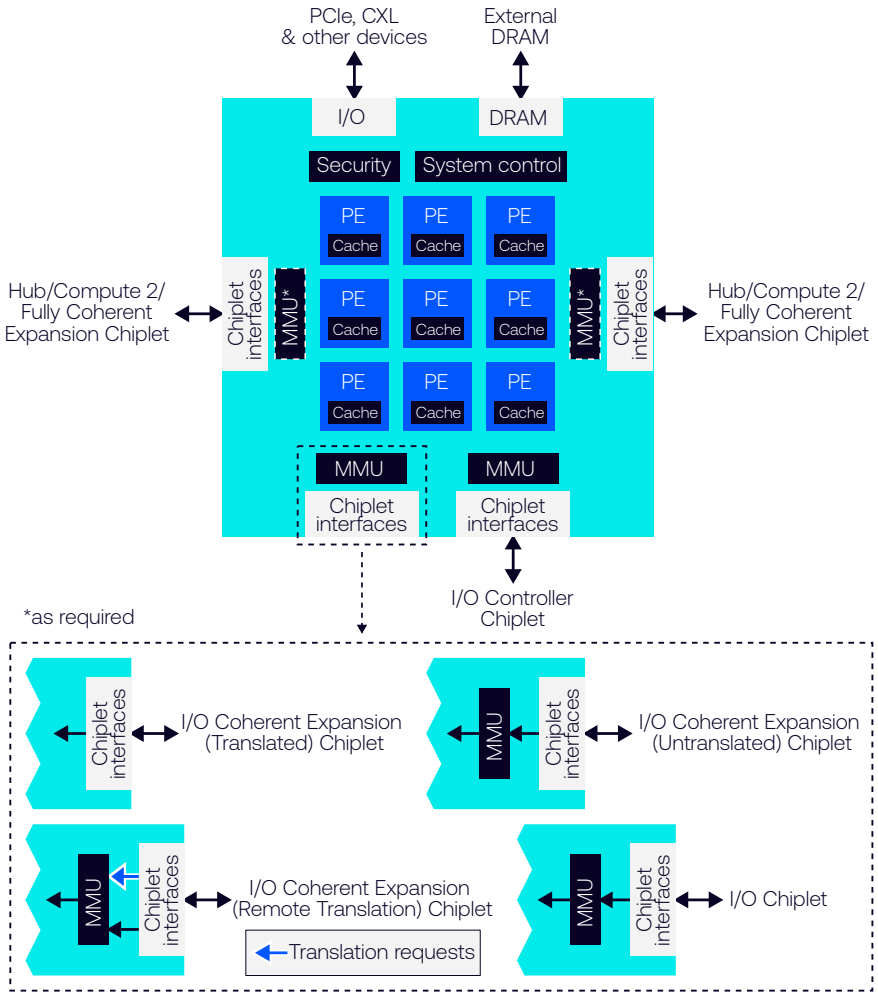

Figure 4.5: Compute 2 Chiplet Example

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 Chiplet Requirements

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

4.3.1.1 Chiplet-to-Chiplet Interfaces

R_{VKGJK} A Compute 2 Chiplet is interfaced with one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet.

(LEVEL-0)

R_{SJNGN} A Compute 2 Chiplet is interfaced with zero or exactly one Hub Chiplet.
(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_DMSQX A Compute 2 Chiplet contains 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 receive 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.

R_CLYNB PEs in a Compute 2 Chiplet access system memory through an MMU. That MMU is contained in the chiplet.

(LEVEL-0)

R_GPSSB I/O coherent or non-coherent agents in a Compute 2 Chiplet access system memory through an MMU. That MMU is contained in the 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.

R$_{TXVCM}$ A Compute 2 Chiplet contains an interrupt controller component.

(LEVEL-0)

I$_{FRVDP}$ Interrupts are received at an interrupt controller 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 interrupt controller component in the chiplet containing the target PE.

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

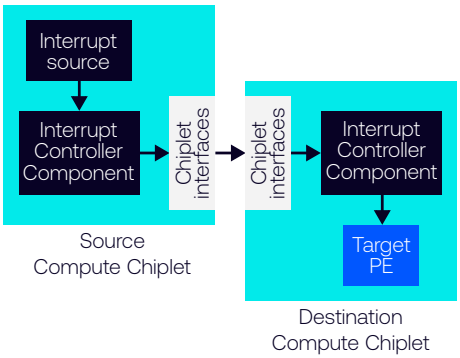

Figure 4.6: Example interrupt communication between two compute chiplets

R$_{THPPG}$ The Compute 2 Chiplet interrupt controller component receives interrupts from within the same chiplet.

(LEVEL-0)

R$_{NYZVM}$ The Compute 2 Chiplet interrupt controller component receives interrupts from other chiplets in the system that do not contain an interrupt controller component.

(LEVEL-0)

R$_{YNMLX}$ The Compute 2 Chiplet interrupt controller component receives interrupts for the PEs it contains forwarded from interrupt controller components in other chiplets.

(LEVEL-0)

R$_{SGZXK}$ The Compute 2 Chiplet interrupt controller component forwards interrupts to a PE, where the target PE is in the same chiplet.

(LEVEL-0)

R$_{VVDFD}$ Where the target PE of an interrupt is in another chiplet, the interrupt controller component in a Compute 2 Chiplet forwards the interrupt to the interrupt controller component in the chiplet of the target PE.

(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 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)

R$_{CNDVJ}$ A Compute 2 Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.3.1.11 Security

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$_{KGFNL}$ 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 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)

4.3.1.13 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 provides the system with 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.

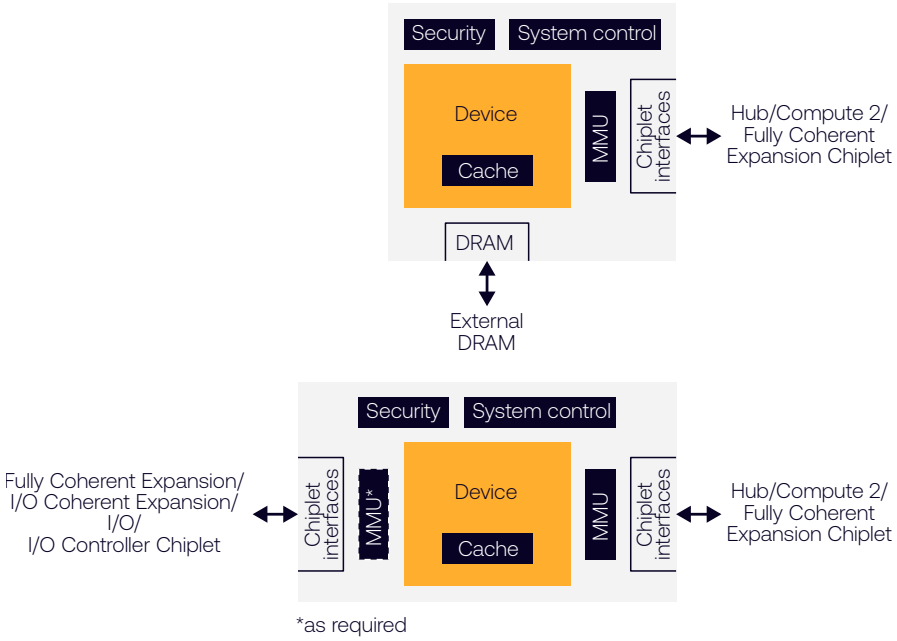

Figure 4.7: Fully Coherent Expansion Chiplet Examples

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet.

(LEVEL-0)

R_{HXGDG} A Fully Coherent Expansion Chiplet is interfaced with zero or exactly one Hub Chiplet.

(LEVEL-0)

R_{GHKCL} A Fully Coherent Expansion Chiplet is interfaced with zero or exactly one 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$_{LVJXD}$ A Fully Coherent Expansion Chiplet might contain system main memory, either integrated or directly connected.

(LEVEL-0)

R$_{XVWQF}$ The system main 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)

R$_{WHJVN}$ Any system addressable memory in the system is available through a Fully Coherent Expansion Chiplet to agents within 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.

R$_{PBNCH}$ I/O coherent or non-coherent agents in a Fully Coherent Expansion Chiplet access system memory through an MMU. That MMU is contained in the chiplet.

(LEVEL-0)

R$_{FDDHQ}$ Fully coherent agents in a Fully Coherent Expansion Chiplet access system memory through an MMU. That MMU is contained in the 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.

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.

R$_{FXKVK}$ A Fully Coherent Expansion Chiplet can contain an interrupt controller component.

(LEVEL-0)

IDYPHT Interrupts are either:

- If the chiplet has an interrupt controller component, received at the Fully Coherent Expansion Chiplet interrupt controller component and then forwarded to the interrupt controller component in the chiplet that contains the target PE.
- If the chiplet does not have an interrupt controller component, forwarded as an MSI to a connected chiplets interrupt controller component.

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

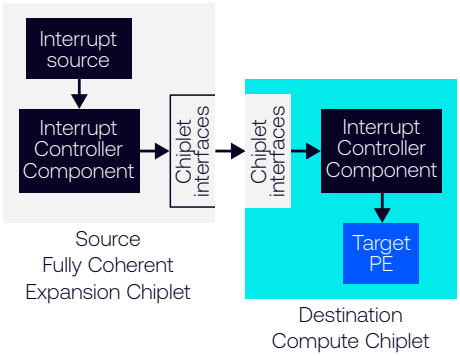

Figure 4.8: Example interrupt communication between fully coherent expansion and compute chiplets

RZRPHF Where a Fully Coherent Expansion Chiplet contains an interrupt controller component, the interrupt controller component receives interrupts from within the same chiplet.

(LEVEL-0)

RQZSYG Where a Fully Coherent Expansion Chiplet contains an interrupt controller component, it can receive interrupts from other chiplets that do not contain an interrupt controller component.

(LEVEL-0)

RPWHNJ Where a Fully Coherent Expansion Chiplet contains an interrupt controller component, the interrupt controller component forwards interrupts to the interrupt controller component in the chiplet of the target PE.

(LEVEL-0)

RRNNVV Where a Fully Coherent Expansion Chiplet does not contain an interrupt controller component, it transmits interrupts as MSIs to an interrupt controller component in a connected chiplet.

(LEVEL-0)

4.4.1.9 Chiplet System Control

RZNXXR A Fully Coherent Expansion Chiplet contains a system controller (see System Controller).

(LEVEL-0)

RRJMNL The system controller in a Fully Coherent Expansion Chiplet is a secondary system controller.

(LEVEL-0)

RXLWKP A Fully Coherent Expansion Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.4.1.10 Security

RFFWWY 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 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)

4.4.1.12 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

IXBDVG The Fully Coherent Expansion (Remote Translation) Chiplet type provides the system with application specific acceleration. The chiplet might include an interface to external memory.

This chiplet type is intended for system specific expansion with complex Fully Coherent devices where the trust boundary for the host chiplet with which it is connected is at the chiplet interface. For example, when integrating third-party solutions. The chiplet is 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.

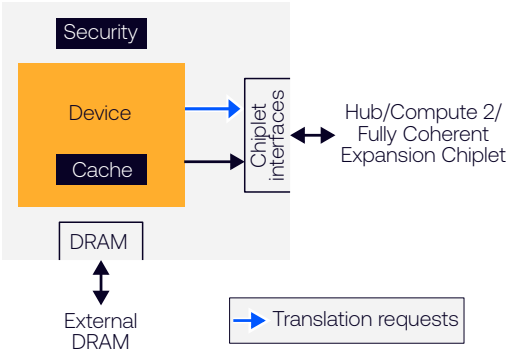

Figure 4.9: Fully Coherent Expansion (Remote Translation) Chiplet Example

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet.

(LEVEL-0)

R_{VZRNQ} A Fully Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or exactly one Hub Chiplet.
(LEVEL-0)

R_{ZKKWN} A Fully Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or exactly one Compute 2 Chiplet.

(LEVEL-0)

R_{VGSNV} A Fully Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or exactly one Fully Coherent Expansion 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 fully 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_{WNHGS} A Fully Coherent Expansion (Remote Translation) Chiplet might contain 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.

R$_{XMDFB}$ I/O coherent or non-coherent agents in a Fully Coherent Expansion (Remote Translation) Chiplet access system memory through an MMU. That MMU is not contained in the chiplet.

(LEVEL-0)

R$_{LMWRD}$ Fully coherent agents in a Fully Coherent Expansion (Remote Translation) Chiplet access system memory through an MMU. That MMU is not contained in the chiplet.

(LEVEL-0)

R$_{DDBSL}$ Any agents in a Fully Coherent Expansion (Remote Translation) Chiplet that request address translation information do so from an MMU. That MMU is not in the 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.

A Fully Coherent Expansion (Remote Translation) Chiplet can be a source of interrupts. These interrupts are forwarded as MSIs.

R$_{PJVLK}$ A Fully Coherent Expansion (Remote Translation) Chiplet transmits interrupts as MSIs to a connected chiplet.

(LEVEL-0)

4.5.1.9 Chiplet System Control

R$_{RVBSS}$ A Fully Coherent Expansion (Remote Translation) Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R$_{CWKWQ}$ The system controller in a Fully Coherent Expansion (Remote Translation) Chiplet is a secondary system controller.

(LEVEL-0)

R$_{NYNCY}$ A Fully Coherent Expansion (Remote Translation) Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.5.1.10 Security

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)

4.5.1.11 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)

4.5.1.12 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

I_NKFQB

The I/O Coherent Expansion (Translated) Chiplet type provides the system with 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.

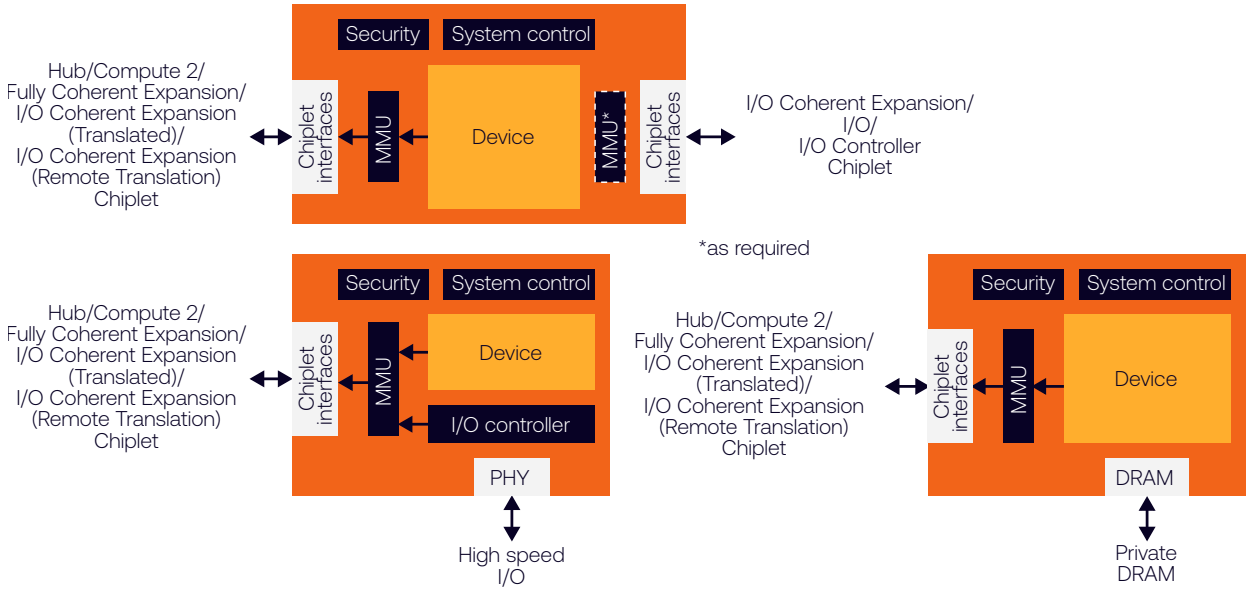

Figure 4.10: I/O Coherent Expansion (Translated) Chiplet Examples

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet, or
- a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

R_{MPKFL} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or exactly one Hub Chiplet.
(LEVEL-0)

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

R_{RWJXM} A I/O Coherent Expansion (Translated) Chiplet is interfaced with zero or exactly one Fully Coherent Expansion Chiplet.
(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_{HKZRJ} See Fully Coherent Expansion Chiplet Interface Requirements.

I_{MTRFG} See I/O Coherent Expansion (Translated) 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_PGQRS A I/O Coherent Expansion (Translated) Chiplet might contain 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_HSWKR Any system addressable memory in the system is available through a I/O Coherent Expansion (Translated) Chiplet to agents within the chiplet or in other chiplets in the system.

(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.

R_XHLRJ I/O coherent or non-coherent agents in a I/O Coherent Expansion (Translated) Chiplet access system memory through an MMU. That MMU is contained in the 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.

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.

R_RDSRT A I/O Coherent Expansion (Translated) Chiplet can contain an interrupt controller component.

(LEVEL-0)

I_TFVYT Interrupts are either:

- If the chiplet has an interrupt controller component, received at the I/O Coherent Expansion (Translated) Chiplet interrupt controller component and then forwarded to the interrupt controller component in the chiplet that contains the target PE.
- If the chiplet does not have an interrupt controller component, forwarded as an MSI to a connected chiplets interrupt controller component.

R_{ZHDSR} Where a I/O Coherent Expansion (Translated) Chiplet contains an interrupt controller component, the interrupt controller component receives interrupts from within the same chiplet.

(LEVEL-0)

R_{LPFHQ} Where a I/O Coherent Expansion (Translated) Chiplet contains an interrupt controller component, it can receive interrupts from other chiplets that do not contain an interrupt controller component.

(LEVEL-0)

R_{SXYNL} Where a I/O Coherent Expansion (Translated) Chiplet contains an interrupt controller component, the interrupt controller component forwards interrupts to the interrupt controller component in the chiplet of the target PE.

(LEVEL-0)

R_{SDNYR} Where a I/O Coherent Expansion (Translated) Chiplet does not contain an interrupt controller component, it transmits interrupts as MSIs to an interrupt controller component 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)

R_{PHQVC} A I/O Coherent Expansion (Translated) Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.6.1.10 Security

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} A trusted security agent in a I/O Coherent Expansion (Translated) Chiplet is a secondary trusted security agent.

(LEVEL-0)

4.6.1.11 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)

4.6.1.12 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

ITLKDT

The I/O Coherent Expansion (Untranslated) Chiplet type provides the system with 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 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.

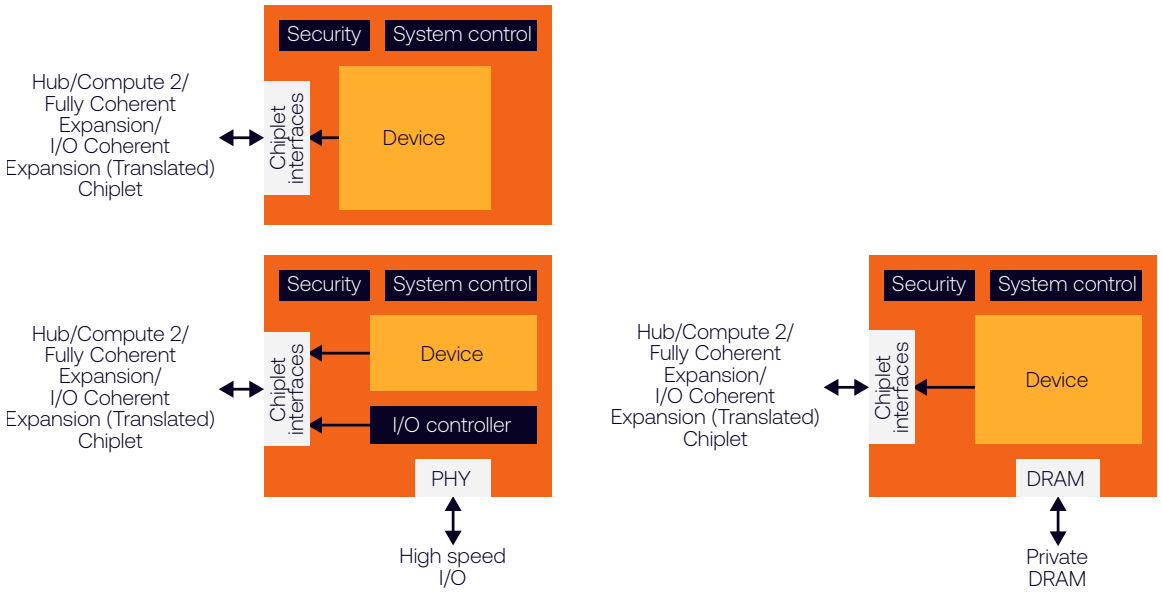

Figure 4.11: I/O Coherent Expansion (Untranslated) Chiplet Examples

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet, or
- a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

R_{GNBSW} A I/O Coherent Expansion (Untranslated) Chiplet is interfaced with zero or exactly one Hub Chiplet.
(LEVEL-0)

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

R_{WFLXB} A I/O Coherent Expansion (Untranslated) Chiplet is interfaced with zero or exactly one Fully Coherent Expansion Chiplet.
(LEVEL-0)

R_{JHQPD} A I/O Coherent Expansion (Untranslated) Chiplet is interfaced with zero or exactly one I/O Coherent Expansion (Translated) Chiplet.
(LEVEL-0)

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

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

R_{PQGLG} I/O coherent or non-coherent agents in a I/O Coherent Expansion (Untranslated) Chiplet access system memory through an MMU. That MMU is not contained in the 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 as MSIs.

R_{RBSQV} A I/O Coherent Expansion (Untranslated) Chiplet transmits interrupts as MSIs to a connected chiplet.

(LEVEL-0)

4.7.1.9 Chiplet System Control

R_{KMTCS} A I/O Coherent Expansion (Untranslated) Chiplet contains one or more system control agents (see System Control Agent).

(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_{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 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)

4.7.1.12 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

IWJTFD

The I/O Coherent Expansion (Remote Translation) Chiplet type provides the system with 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.

This chiplet type is intended for system specific expansion with complex I/O Coherent devices where the trust boundary for the host chiplet with which it is connected is at the chiplet interface. For example, when integrating third-party solutions. The chiplet is 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.

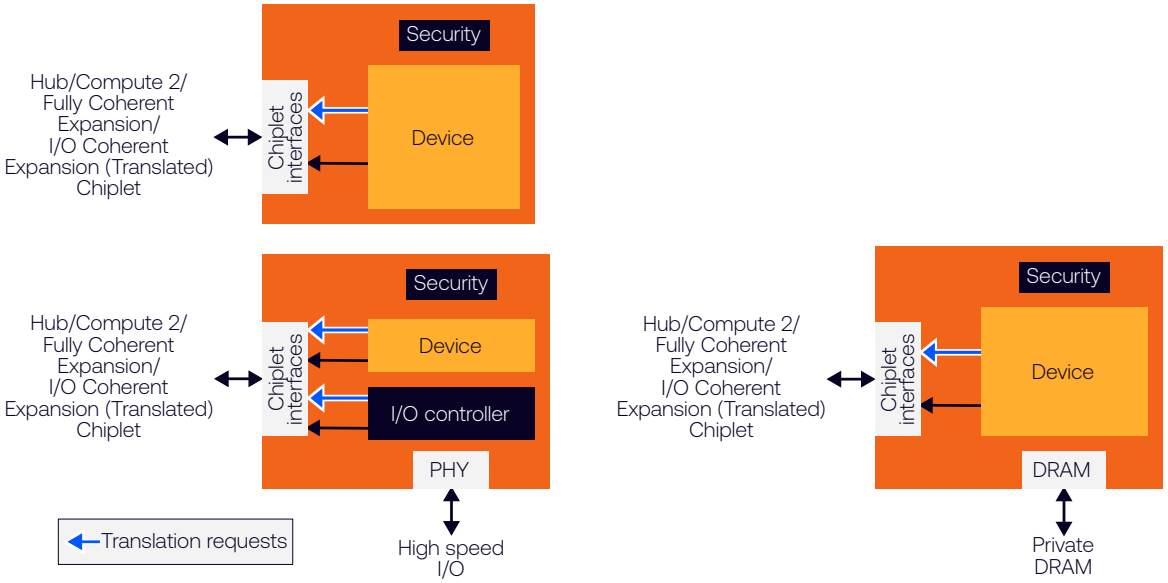

Figure 4.12: I/O Coherent Expansion (Remote Translation) Chiplet Examples

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet, or
- a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

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

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

R_{LVZJJ} A I/O Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or exactly one Fully Coherent Expansion Chiplet.

(LEVEL-0)

R_{YKSJV} A I/O Coherent Expansion (Remote Translation) Chiplet is interfaced with zero or exactly one I/O Coherent Expansion (Translated) 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_{GXVLS} A I/O Coherent Expansion (Remote Translation) Chiplet might contain 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.

R_GNNND I/O coherent or non-coherent agents in a I/O Coherent Expansion (Remote Translation) Chiplet access system memory through an MMU. That MMU is not contained in the chiplet.

(LEVEL-0)

R_JWZKZ Any agents in a I/O Coherent Expansion (Remote Translation) Chiplet that request address translation information do so from an MMU. That MMU is not in the 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 connected chiplet.

(LEVEL-0)

4.8.1.9 Chiplet System Control

R_XCBFD A I/O Coherent Expansion (Remote Translation) Chiplet contains a system controller (see System Controller).

(LEVEL-0)

R_RHJMG The system controller in a I/O Coherent Expansion (Remote Translation) Chiplet is a secondary system controller.

(LEVEL-0)

R_PWMCT A I/O Coherent Expansion (Remote Translation) Chiplet contains one or more system control agents (see System Control Agent).

(LEVEL-0)

4.8.1.10 Security

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)

4.8.1.11 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)

4.8.1.12 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 provides the system with 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.

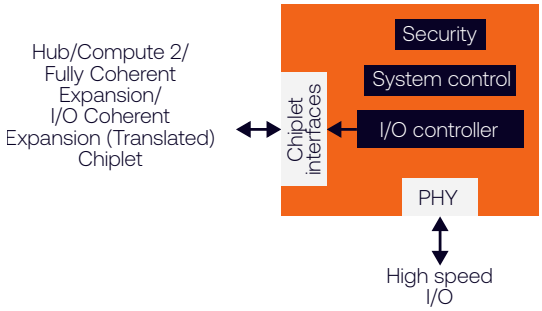

Figure 4.13: I/O Chiplet Example

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet, or
- a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

R_{WWGKL} A I/O Chiplet is interfaced with zero or exactly one Hub Chiplet.

(LEVEL-0)

R_{HCYCZ} A I/O Chiplet is interfaced with zero or exactly one Compute 2 Chiplet.

(LEVEL-0)

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

(LEVEL-0)

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

(LEVEL-0)

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

I_{KWXNR} See I/O 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_{RTPGY} A I/O Chiplet might contain 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.

R_MGCKS I/O coherent or non-coherent agents in a I/O Chiplet access system memory through an MMU. That MMU is not contained in the 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 on a I/O Chiplet can generate interrupts. These interrupts are forwarded as MSIs.

4.9.1.9 Chiplet System Control

R_LCHML A I/O Chiplet contains one or more system control agents (see System Control Agent).

(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_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 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)

4.9.1.12 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-2 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.

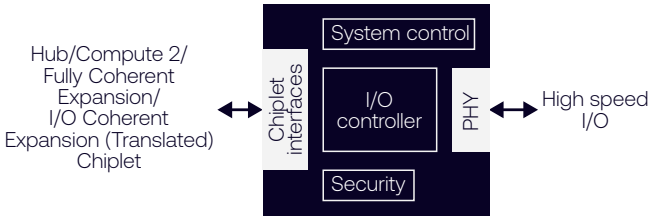

Figure 4.14: I/O Controller Chiplet Example

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 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 one of the following (mutually exclusive):

- a Hub Chiplet, or
- a Compute 2 Chiplet, or
- a Fully Coherent Expansion Chiplet, or
- a I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

R_{LTWZF} A I/O Controller Chiplet is interfaced with zero or exactly one Hub Chiplet.

(LEVEL-0)

R_{JWNXJ} A I/O Controller Chiplet is interfaced with zero or exactly one Compute 2 Chiplet.

(LEVEL-0)

R_{RDQYR} A I/O Controller Chiplet is interfaced with zero or exactly one Fully Coherent Expansion Chiplet.

(LEVEL-0)

R_{DSBQD} A I/O Controller Chiplet is interfaced with zero or exactly one I/O Coherent Expansion (Translated) Chiplet.

(LEVEL-0)

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

I_{CPZPP} See I/O Controller 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_{QWLMP} A I/O Controller Chiplet might contain 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.

R_{QMDXK} I/O coherent or non-coherent agents in a I/O Controller Chiplet access system memory through an MMU. That MMU is not contained in the 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 on a I/O Controller Chiplet can generate interrupts. These interrupts are forwarded as MSIs.

R_{MDBRR} A I/O Controller Chiplet transmits interrupts as MSIs to a connected chiplet.

(LEVEL-0)

4.10.1.9 Chiplet System Control

R_{KZQVC} A I/O Controller Chiplet contains one or more system control agents (see System Control Agent).

(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_{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 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)

4.10.1.12 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)

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 FCSA 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.

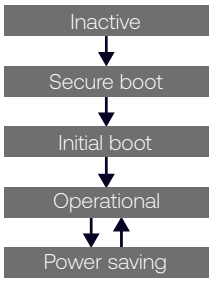

Figure 5.1: Chiplet Activity States

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 code 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 Code 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
Trusted Security Agent Communication	Unavailable	Available	Available	Available	Available
System Control	Unavailable	Available	Available	Available	Available
Debug Access	Unavailable	Available	Available	Available	Available

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.

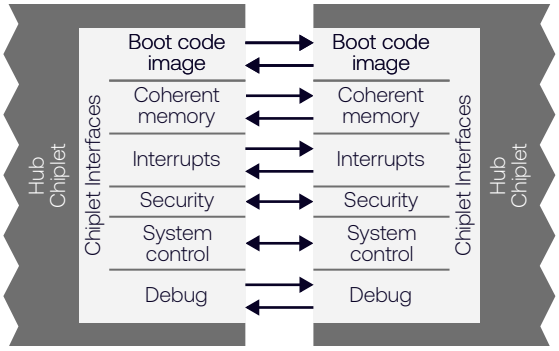

Figure 5.2: Hub Chiplet to Hub Chiplet Interfaces

5.2.1 Boot Code Image Load Interface

I$_{GNTLV}$ A Boot Code 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 code images and the other chiplet is a destination. There is a Boot Code Image Load Interface between a Hub Chiplet and another Hub Chiplet that is directly connected, that is capable of transporting boot code images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-1)

R$_{KMPRP}$ A Boot Code 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-1)

U$_{CGLNT}$ For example, a Boot Code 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 Code 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.

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 and non-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 Interrupt Handling

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

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 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.

5.2.5 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.

5.2.6 Debug

5.2.6.1 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.

5.3 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.

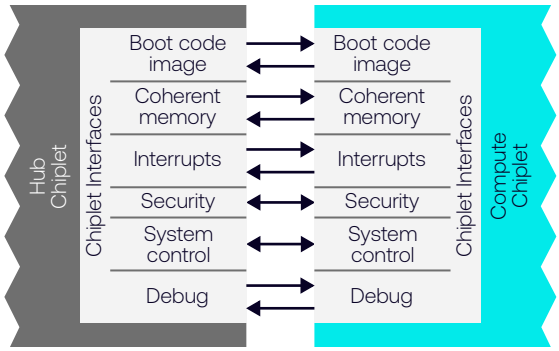

Figure 5.3: Hub Chiplet to Compute Chiplet Interfaces

5.3.1 Boot Code Image Load Interface

I_{ZSFTB} Non-volatile memory can be accessed by a Hub Chiplet. A Boot Code 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 code images and the other chiplet is a destination. There is a Boot Code Image Load Interface between a Hub Chiplet and a compute chiplet that is directly connected, that is capable of transporting boot code images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-1)

R_{YWSTJ} A Boot Code 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-1)

U_{CCDDY} For example, a Boot Code 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 Code 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.

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.

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 and non-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 Interrupt Handling

I$_{FGYNK}$ This section concerns interrupts that are targeted at application PEs running the main operating system.

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 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.

5.3.5 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.

5.3.6 Debug

5.3.6.1 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.

5.4 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.

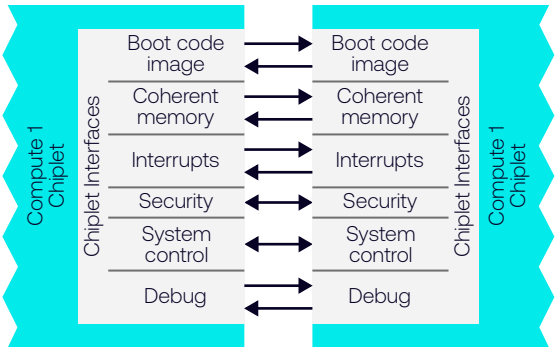

Figure 5.4: Compute 1 Chiplet to Compute 1 Chiplet Interfaces

5.4.1 Boot Code Image Load Interface

I_FZTRR A Boot Code 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 code images and the other chiplet is a destination. There is a Boot Code Image Load Interface between a Compute 1 Chiplet and another Compute 1 Chiplet that is directly connected, that is capable of transporting boot code images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-1)

R_RSGWB A Boot Code 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-1)

U_LMNCN For example, a Boot Code 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 Code 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.

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 and non-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 Interrupt Handling

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

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 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, that transports messages between the trusted security agents in 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.

5.4.5 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, that transports messages between the system controllers in 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.

5.4.6 Debug

5.4.6.1 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.

5.5 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.

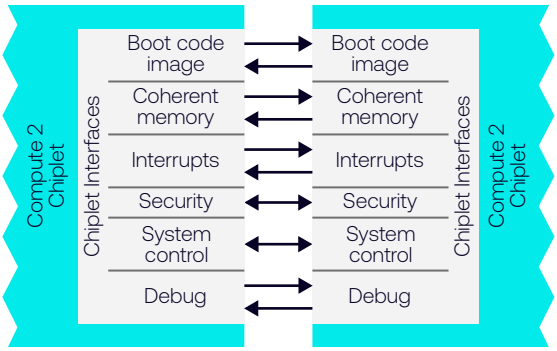

Figure 5.5: Compute 2 Chiplet to Compute 2 Chiplet Interfaces

5.5.1 Boot Code Image Load Interface

I$_{SBNTD}$ A Boot Code 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 code images and the other chiplet is a destination. There is a Boot Code Image Load Interface between a Compute 2 Chiplet and another Compute 2 Chiplet that is directly connected, that is capable of transporting boot code images from the source chiplet to memory locations in the destination chiplet, or in other chiplets connected to that destination chiplet.

(LEVEL-1)

R$_{RLQYC}$ A Boot Code 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-1)

U$_{VSRWY}$ For example, a Boot Code 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 Code 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.

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 and non-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 Interrupt Handling

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

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 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.

5.5.5 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.

5.5.6 Debug

5.5.6.1 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.

5.6 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.

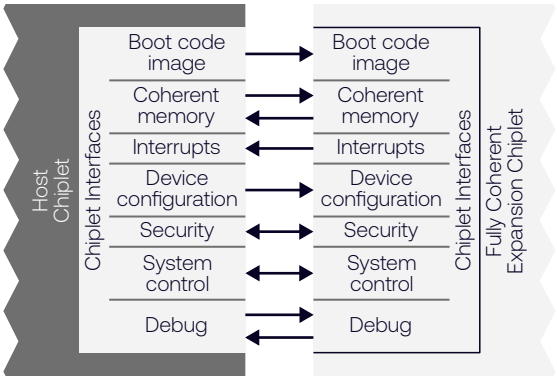

Figure 5.6: Fully Coherent Expansion Chiplet Interfaces

5.6.1 Boot Code Image Load Interface

I_{YDTNL} Non-volatile memory can be accessed by a host chiplet. A Boot Code Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{PRKPY} There is a Boot Code Image Load Interface between a host chiplet and a Fully Coherent Expansion Chiplet that is directly connected, that is capable of transporting boot code images from a host chiplet to memory locations in a Fully Coherent Expansion Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-1)

R_{WPZNH} A Boot Code 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-1)

U_{LQCLJ} For example, a Boot Code 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 Code 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.

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.

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 and non-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 Interrupt Handling

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

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 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.

5.6.5 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.

5.6.6 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.

5.6.7 Debug

5.6.7.1 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.

5.7 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.

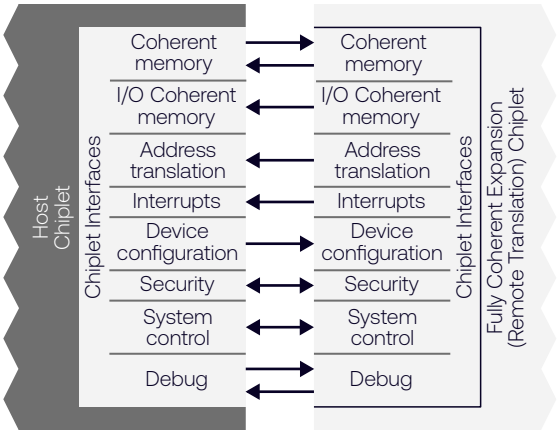

Figure 5.7: Fully Coherent Expansion (Remote Translation) Chiplet Interfaces

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.

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.

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 and non-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 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 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 Interrupt Handling

I$_{ZHFTY}$ This section concerns interrupts that are targeted at application PEs running the main operating system.

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.

(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.

5.7.5 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.

5.7.6 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.

5.7.7 System Control Communication

R$_{GQNNR}$ There is a System Control Interface between a host chiplet and a Fully Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R$_{XXTQB}$ A System Control 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$_{MWCKT}$ For example, a System Control 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$_{DKNSC}$ For implementation details regarding this interface see System Controller Communication.

5.7.8 Debug

5.7.8.1 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.

5.8 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.

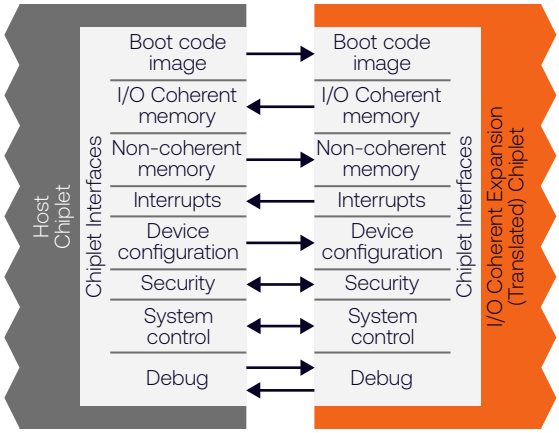

Figure 5.8: I/O Coherent Expansion (Translated) Chiplet Interfaces

5.8.1 Boot Code Image Load Interface

I_{SLXGZ} Non-volatile memory can be accessed by a host chiplet. A Boot Code Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{CNCGH} There is a Boot Code 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 code 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-1)

R_{JLGGN} A Boot Code 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-1)

U_{VRZYT} For example, a Boot Code 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 Code 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.

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 MSIs.

(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) 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 Debug

5.8.8.1 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.

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

I$_{GZLLQ}$ 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.

I$_{QGNZM}$ An overview of the interfaces between the two chiplets is depicted in Figure 5.9.

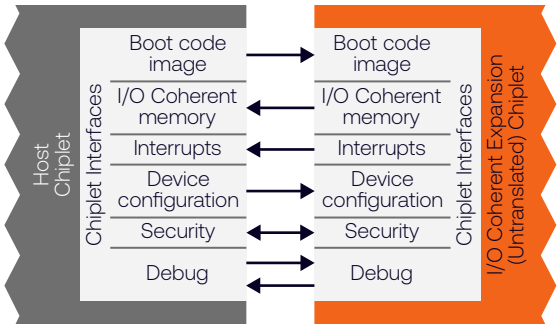

Figure 5.9: I/O Coherent Expansion (Untranslated) Chiplet Interfaces

5.9.1 Boot Code Image Load Interface

I$_{MHJVS}$ Non-volatile memory can be accessed by a host chiplet. A Boot Code Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R$_{WBSBR}$ There is a Boot Code 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 code 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-1)

R$_{WWJWH}$ A Boot Code 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-1)

U$_{NMRMC}$ For example, a Boot Code 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.

I$_{DXGYQ}$ For implementation details regarding this interface see Boot Code 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.

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 in the 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 Debug

5.9.6.1 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.

5.10 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.

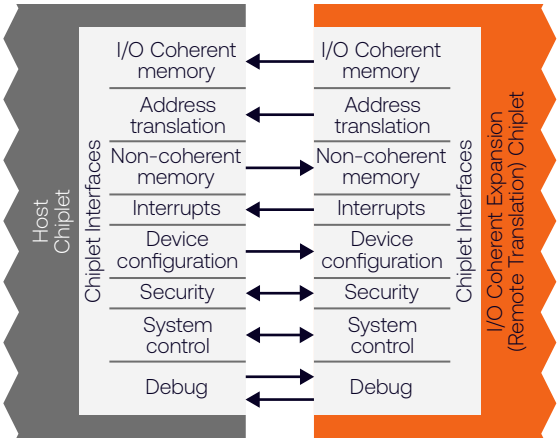

Figure 5.10: I/O Coherent Expansion (Remote Translation) Chiplet Interfaces

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 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.

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.

(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 Control Communication

R_{QXZDW} There is a System Control Interface between a host chiplet and a I/O Coherent Expansion (Remote Translation) Chiplet that is directly connected, that transports messages between the system controllers in the chiplets.

(LEVEL-0)

R_{VPNZQ} A System Control 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_{XWMWN} For example, a System Control 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_{YKBDB} For implementation details regarding this interface see System Controller Communication.

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.

5.11 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.

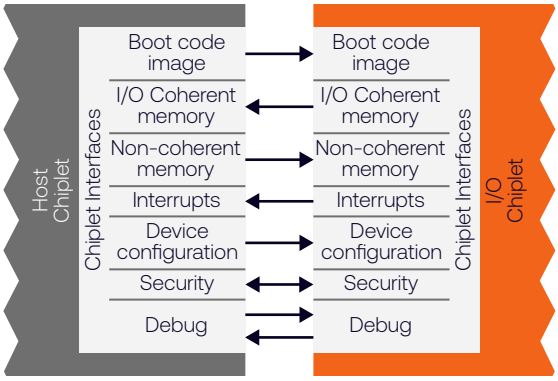

Figure 5.11: I/O Chiplet Interfaces

5.11.1 Boot Code Image Load Interface

I_{HFCCM} Non-volatile memory can be accessed by a host chiplet. A Boot Code Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{WGMMT} There is a Boot Code Image Load Interface between a host chiplet and a I/O Chiplet that is directly connected, that is capable of transporting boot code images from a host chiplet to memory locations in a I/O Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-1)

R_{GBYYK} A Boot Code 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-1)

U_{QPMHQ} For example, a Boot Code 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 Code 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.

I$_{XYFWM}$ A I/O Chiplet might contain components that generate interrupts.

R$_{DFQYV}$ 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 Debug

5.11.7.1 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.

5.12 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.

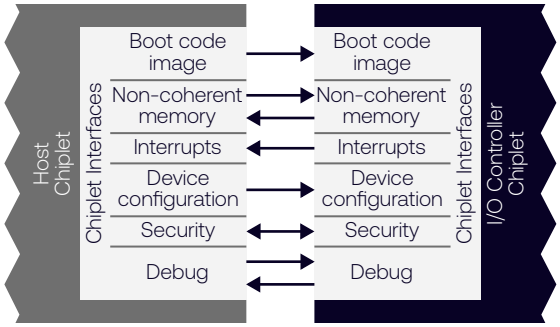

Figure 5.12: I/O Controller Chiplet Interfaces

5.12.1 Boot Code Image Load Interface

I_{JBQQV} Non-volatile memory can be accessed by a host chiplet. A Boot Code Image Load Interface transports boot code images from the non-volatile memory to each chiplet in the system.

R_{RDVNB} There is a Boot Code Image Load Interface between a host chiplet and a I/O Controller Chiplet that is directly connected, that is capable of transporting boot code images from a host chiplet to memory locations in a I/O Controller Chiplet, or in other chiplets connected to that chiplet.

(LEVEL-1)

R_{JHJVP} A Boot Code 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-1)

U_{FWTNS} For example, a Boot Code 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 Code 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.

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 in the 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 Debug

5.12.6.1 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.

6
Common Chiplet Components

Some chiplet components are common to the specifications of several FCSA chiplet types. These components are described in this chapter.

The components are:

- System Control Framework.
- System Control Agent.
- Trusted Security Agent.

6.1 System Control Framework

DQGXRY

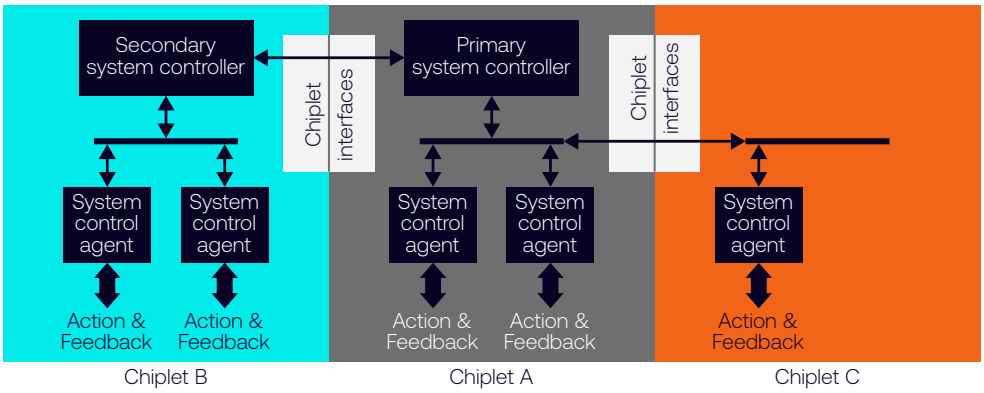

Figure 6.1: System Control Framework Example

The System Control Framework is used for low-level SoC specific management tasks such as power and thermal management.

The system control functionality is distributed over multiple components and multiple chiplets, as depicted in Figure 6.1. There are two types of component:

- System Controller.
- System Control Agent.

There is typically a *system controller* component in each chiplet, although there can be more than one in a chiplet and some chiplet types do not require one at all. 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.

A system controller sets system control policy within the scope of its action and this policy is effected directly or locally by one or more *system control agents*.

6.1.1 System Controller

DXJKRK A System Controller is a component of the System Control Framework that makes system control policy decisions within a scope of influence in the system.

A system controller is a complex component that includes one or more PEs and supporting components and that executes firmware. It might include components for I/O and security.

RTMGXQ There is only one *primary* system controller in the system.

(LEVEL-0)

RNHBRK The primary system controller coordinates the actions of the secondary system controllers.

(LEVEL-0)

DBTCVH 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 Code Image Load Interface.

(LEVEL-1)

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 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 of the System Control Framework 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. It might include automatic functionality that enables real-time actions in response to feedback.

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)

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.

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 Code Image Load Interface.

(LEVEL-1)

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 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)

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 FCSA does not define any new interface standards. Where applicable standards are not available for an interface the transport and protocol are TO BE DEFINED.

I_{KPDLW} The protocols and transports that the FCSA 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 FCSA, 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 FCSA, might include features that support meeting system-level requirements that concern hardware faults. This includes interfaces that use the UCIe [4] and I3C [5] 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 Code Image Load Interface

I_{FXRVR} Boot code images for certain components in the system are required to be loaded across chiplet boundaries.

R_{KJVCD} The protocol for the Boot Code Image Load Interface is TO BE DEFINED.

(LEVEL-FULL)

U_{PDHSQ} The Boot Code 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 Distributed Virtual Memory (DVM) messages.

R_{JCLGK} The protocol for the Coherent Memory Traffic Interface is AMBA CHI Chip-to-Chip (CHI C2C) [6].

(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 TO BE 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 TO BE DEFINED.

(LEVEL-1)

U_{LVWDK} A protocol option for the I/O Coherent Memory Traffic Interface is AMBA CHI Chip-to-Chip (CHI C2C) [6].

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 TO BE DEFINED.

(LEVEL-1)

U_{HHHBY} A protocol option for the Non-Coherent Memory Traffic Interface is AMBA CHI Chip-to-Chip (CHI C2C) [6].

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 TO BE DEFINED.

(LEVEL-1)

U_MDRHJ A protocol option for the Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet is AMBA CHI Chip-to-Chip (CHI C2C) [6].

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 TO BE 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 TO BE DEFINED.

(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 TO BE DEFINED.

(LEVEL-FULL)

7.2.8 Interrupt Handling

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

R_{RJNQG} Interrupts are sent as MSI messages.

(LEVEL-FULL)

R_{CWKYG} Additionally, interrupts can be forwarded between interrupt controller components using memory-mapped accesses.

(LEVEL-FULL)

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 TO BE DEFINED.

(LEVEL-FULL)

U_{XDGJH} If the Boot Code 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 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 TO BE DEFINED.

(LEVEL-FULL)

U_{VSLJT} The protocol options for the System Control Interface include the System Control and Management Interface (SCMI) [7] and the RISC-V Platform Management Interface (RPMI) [8].

7.2.11 Debug

7.2.11.1 Debug Access Interface

R_{LXCQZ} Debug access protocol to a chiplet is provided by JTAG IEEE Std 1149.

(LEVEL-FULL)

7.3 Interface Transports

This section describes the interface transports for the specified interfaces.

7.3.1 Boot Code Image Load Interface

I_{LHQXN} Boot code images for certain components in the system are required to be loaded across chiplet boundaries.

R_{GJBYZ} The transport for the Boot Code Image Load Interface is TO BE DEFINED.

(LEVEL-FULL)

U_{GDGJF} A recommended transport for the Boot Code Image Load Interface is an I3C interface [5].

U_{GSFJN} The Boot Code 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 Distributed Virtual Memory (DVM) messages.

R_{ZZXCL} The transport for the Coherent Memory Traffic Interface is TO BE DEFINED.

(LEVEL-FULL)

U_{LKFMS} The transport options for the Coherent Memory Traffic Interface include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{YVNJF} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(LEVEL-FULL)

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 TO BE DEFINED.

(LEVEL-FULL)

U_{SGMNK} The transport options for the Untranslated I/O Coherent Memory Traffic Interface include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{CVVHX} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(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 TO BE DEFINED.

(LEVEL-FULL)

U_{DSWDX} The transport options for the I/O Coherent Memory Traffic Interface include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{NFWLC} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(LEVEL-FULL)

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 TO BE DEFINED.

(LEVEL-FULL)

U_{DLZKL} The transport options for the Non-Coherent Memory Traffic Interface include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{JQBWY} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(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 TO BE DEFINED.

(LEVEL-FULL)

U_{XQHWQ} The transport options for the Device Configuration Interface for a I/O Coherent Expansion (Translated) Chiplet include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{MRQTT} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(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 TO BE DEFINED.

(LEVEL-FULL)

U_{NBRSY} The transport options for the Device Configuration Interface for a I/O Coherent Expansion (Untranslated) Chiplet include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{WRTTH} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(LEVEL-FULL)

7.3.6.3 Device Configuration Interface for an 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 TO BE DEFINED.

(LEVEL-FULL)

U_{SZTZW} The transport options for the Device Configuration Interface for a I/O Coherent Expansion (Remote Translation) Chiplet include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{VZYTP} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(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 TO BE DEFINED.

(LEVEL-FULL)

U_{QNVFW} The transport options for the Address Translation Interface for a I/O Coherent Expansion (Remote Translation) Chiplet include the UCIe v1.1 [4] streaming protocol and the OCP Bunch of Wires (BoW) [9] protocol.

(LEVEL-FULL)

U_{NXBZH} A more recent version of the protocol specified can be used where that version is backwards compatible with the version specified.

(LEVEL-FULL)

7.3.8 Interrupt Handling

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

R_{ZRGXC} Interrupts are sent as MSI messages.

(LEVEL-FULL)

R_{HSZXD} Additionally, interrupts can be forwarded between interrupt controller components using memory-mapped accesses.

(LEVEL-FULL)

R_{BGHPZ} Interrupts are transported using one of the following depending on chiplet type:

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 TO BE DEFINED.

(LEVEL-FULL)

U_{DNJVQ} A recommended transport for the Trusted Security Agent Communication Interface is a dedicated I3C interface [5].

X_{LPGMG} The I3C interface [5] 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 Code 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)

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 TO BE 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 [5].

X_{ZPTVP} The I3C interface [5] is a standard, simple interface that is a fit for the purpose of transporting the System Control Interface.

U_{NLWKD} If the Boot Code 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 they can enter power modes where the System Control Interface is not available, such as the *Power saving* chiplet activity state.

7.3.11 Debug

7.3.11.1 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.

A
OCP Checklist

Item	Status or Details	Link to detailed explanation
Is this contribution entered into the OCP Contribution Portal?	Yes	
Was it approved in the OCP Contribution Portal?	Yes	
Is there a Supplier(s) that is building a product based on this Spec? (Supplier must be an OCP Solution Provider)	Yes	
Will Supplier(s) have the product available for GENERAL AVAILABILITY within 120 days?		

Glossary

AMBA

Advanced Microprocessor Bus Architecture - Arm Interconnect and related standards.

ATS

(PCIe) Address Translation Services, see [10].

AXI

Advanced eXtensible Interface, part of the AMBA family of protocols, see [11].

BoW

Bunch of Wires, an OCP PHY, transaction, and link-layer specification for chip and chiplet communications.

C2C

(AMBA) Chip-to-Chip Interface - for transporting AMBA protocols between chiplets. For more information see [6].

CHI

(AMBA) Coherent Hub Interface, for more information see [3].

CXL

Compute eXpress Link, see [12].

DVM

Distributed Virtual Memory.

HES

Hardware Enforced Security.

I3C

Improved Inter-Integrated Circuit (Interface), see [5].

IEEE

Institute of Electrical and Electronics Engineers.

JTAG

Joint Test Action Group.

MHU

Message Handling Unit.

MMU

Memory Management Unit.

MSI

Message Signalled Interrupt.

OCP

Open Compute Project.

PA

Physical Address, an address that identifies a location in the physical memory map.

PCIe

Peripheral Component Interconnect Express, see [10].

PE

Processing Element, typically a CPU.

RFQ

Request for Quote.

RoT

Root of Trust.

RPMI

RISC-V Platform Management Interface Specification [8].

SCMI

System Control and Management Interface, see [7].

SoC

System-on-Chip.

TAP

Test Access Port, part of the JTAG / *IEEE Std 1149.1™-2001* standard, see [13].

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.

TMS

Test Mode Select, a standard JTAG interface signal.

TO BE DEFINED

A feature for which there is an intent to define the implementation in a future version of the FCSA.

UCIe

Universal Chiplet Interface Express, for more information see [4].