IO Remapping Table

Platform Design Document

Non-confidential

Document number: ARM DEN 0049E.d

About this Document
System Software on Arm

Release information

The Change History table lists the changes made to this document.

Table 1-1 Change history

Date	Issue	Confidentiality	Change
17 April 2015	A	Non-Confidential	First release.
1 October 2015	B	Non-Confidential	Added SMMUv3
15 May 2017	C	Non-Confidential	Fixed support for MSIs in SMMUv3 added PMCG
March 2018	D	Non-Confidential	PMCG page 1 support. SMMUv3 fix on proximity node. PASID width added to named nodes. DMA mask added to root complex node.
July 2020	E	Non-Confidential	Bumped up the revisions of the table header, the RC node, the named component node and the SMMU nodes. Added an Identifier field in the node descriptors to aid table cross-referencing. Added the Reserved Memory Range (RMR) node. Tightened the language on permitted ID mappings for certain nodes such as the SMMU node. Introduced a flag in the RC node to express support for PRI. Updated Figure 2 to reflect the right Device IDs being output from SMMU 0 (since NIC 0 does not generate an MSI as per the example). Added a note on use of IMP_DEF input IDs in ID mappings within Named Component nodes.
November 2020	E.a	Non-confidential	Deprecated the Revision field in individual nodes. The table header Revision is the one and only Revision indicator. Any change within the table must result in an update to the header revision. This simplifies software support. Bumped up the Identifier field of the nodes to 32 bits, to establish overall consistency, and parity with existing ACPI _UID definitions. Added a flag called DMAO, and clarifying text to explicitly call out the relation between memory access size limitations and related properties defined in the _DMA object of a named component that is a bridge, with the

			device memory address size limit property of the device defined in the Named Component node. Added a Flags field to the RMR node, with a flag that specifies whether it is permitted for the OS to remap the reserved memory ranges at runtime. Added a clause in the RMR node to mandate the use of the PCI _DSM method for boot configuration, to allow live devices using reserved memory regions to remain functionally operational during OS boot. Added clarifying language on the usage of the COHACC and HTTU override bits in the SMMUv3 node. Added a flag in the Root complex node to declare support for forwarding PASID information to the SMMU on translated transactions.
February 2021	E.b	Non-confidential	Reintroduced the Revision field within IORT node descriptors to support legacy dependencies. Moved flags field in RMR node to the node-specific section to maintain consistency with other nodes and the generic IORT node descriptor format. Deprecated the DMAO flag and replaced it with a clarifying note. Added a clarifying note on the use of PCIe _DSM Function 5 for continuity of operation. Minor fixes to the RMR node example in Appendix A. Notably, NIC 1 is replaced with NIC 0 to match illustration. Also, updated the terminology used in the examples to match definitions in the main section of the specification, notably the ID mapping structures.
January 2022	E.c	Non-confidential	Added descriptor in the root complex node for specifying PASID width supported by the root complex. Added note in the root complex node description that allows the node to view a root complex as a PCIe domain that is not necessarily a physical root complex. This allows PCIe EPs to be variously configured with different properties. Removed restriction on number of Stream IDs that can be associated with memory ranges in an RMR node. Updated Figure 1 and Figure 3 to reflect the fact that it is permissible to have multiple Stream IDs associated with a single RMR node.

			Added clarity on correlation between _DMA methods on named components and root complexes and how they relate to the “Device memory address size limit” and “Memory address size limit” fields of these nodes, respectively. Introduced memory access attributes in the the RMR node.
February 2022	E.d	Non-confidential	Readjusted fields in the RMR node to preserve its original length. This is to ensure backward compatibility with existing software developed for version E.b. Note: Issue E.c is deprecated and must not be referenced.

Arm Non-Confidential Document Licence ("Licence")

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

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

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

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

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

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

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

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

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

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

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE'S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE

DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

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

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

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

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

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. No licence, express, implied or otherwise, is granted to Licensee under this Licence, to use the Arm trade marks in connection with the Document or any products based thereon. Visit Arm's website at https://www.arm.com/company/policies/trademarks for more information about Arm's trademarks.

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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-21585 version 4.0

Contents

1 About this Document

This document provides a proposal for an ACPI representation of IO topology to be used by Arm-based systems.

1.1 References

This document refers to the following documents.

Reference	Document Number	Title
[ACPI]	ACPI	Advanced Configuration and Power Interface Specification
[GIC]	ARM IHI 0069	Arm Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0
[SBSA]	ARM DEN 0029	Server Base System Architecture
[SMMUv2]	ARM IHI 0062	Arm System Memory Management Unit Architecture Specification, SMMU architecture version 2.0
[SMMUv3]	ARM IHI 0070	Arm System Memory Management Unit Architecture Specification, SMMU architecture version 3.0 and version 3.1
[PCIFW]		PCI™ Firmware Specification, Revision 3.1

1.2 Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
RID	Requestor ID for a PCI express device.
BDF	Bus Device Function. Equivalent to a RID.
DeviceID	Identifier for a device exposed to the GICv3/4 Interrupt Translation Service. See [GIC] for more details.
IOVA	Input Output Virtual Address. The virtual addresses seen by IO devices.
ITS	GIC Interrupt Translation Service. See [GIC] for more details.
StreamID	A StreamID uniquely identifies to an SMMU a stream of transactions that can originate from one or more devices but are associated with the same context. See [SMMUv2] for more details.
SBSA	Server Base System Architecture.
IO Coherent	A device is IO Coherent with the processor caches if its transactions snoop the processor caches for cacheable regions of memory. The processor does not snoop the device cache.

1.3 Feedback

Arm welcomes feedback on its documentation.

1.3.1 Feedback on this manual

If you have comments on the content of this manual, send an email to errata@arm.com. Give:

- The title.
- The document and version number, ARM DEN 0049E.d.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

2 Introduction

This document describes the *Input Output Remapping Table* (IORT), which represents the IO topology of an Arm-based system for use with the *Advanced Configuration and Power Interface* (ACPI). The IORT describes how various components are connected together, and how those components that need identification reserve values in the appropriate identification space.

In particular, the IORT:

- Provides an ACPI description for IO topology, SMMUs, and GIC ITSs.
- Identifies which components are behind which SMMU.
- Identifies which components are behind an ITS or group of ITSs.
- Describes the IO relationships of PCIe root complexes and relates this description to the MCFG table [PCIFW] and the ACPI namespace.
- Describes the IO relationships between devices represented in the ACPI namespace.
- Represents the following ID mapping relationships:
    - From BDF requestor ID, for a PCIe device, to a StreamID for an SMMU, and then to a DeviceID for an ITS.
    - From BDF requestor ID to a DeviceID, for a device that is not connected to SMMU but which can generate MSIs.
    - Individual endpoint StreamIDs.
    - Individual endpoint DeviceIDs.
- Does not support arbitrarily complex ID mappings. Only simple offset-based mappings are supported.

The next section describes the IORT in detail. Appendix A explains how the IORT is used to describe a system and provides examples. Finally, Appendix B describes how the memory attributes that are described for a device can be used to ascertain when cache management is required.

3 IO Remapping Table

Table 2 shows the structure of the IORT. Apart from the basic header, the table contains a number of IORT Nodes. Each node represents a component, which can be an SMMU, an ITS Group, a root complex, or a component that is described in the namespace.

Table 2 The IORT

Field	Byte Length	Byte Offset	Description
Header			Standard ACPI format for header.
Signature	4	0	'IORT'. IO Remapping Table.
Length	4	4	Length, in bytes, of the entire IORT.
Revision	1	8	5
Checksum	1	9	The entire table must sum to zero.
OEMID	6	10	OEM ID.
OEM Table ID	8	16	For the IORT, the table ID is the manufacture model ID.
OEM Revision	4	24	OEM revision of the IORT for the supplied OEM Table ID.
Creator ID	4	28	The vendor ID of the utility that created the table. For tables containing Definition Blocks, this is the ID for the ASL Compiler.
Creator Revision	4	32	The revision of the utility that created the table. For tables containing Definition Blocks, this is the revision for the ASL Compiler.
Body			
Number of IORT Nodes	4	36	The number of nodes in the IORT Node Array.
Offset to Array of IORT Nodes	4	40	The offset from the start of the table to the first node in the array of IORT nodes.
Reserved	4	44	Reserved, must be zero.
Optional padding	---	--	
Array of IORT Nodes	---	--	Array of IORT Nodes.

Nodes describe component *identifiers* (IDs) and the mapping between the input space of those IDs and the output space. For example, a root complex node, sitting behind an SMMU, describes how the input RID space of the root complex maps onto the output StreamID space that is directed to the SMMU. Not every node needs IDs. Some nodes, such as those that represent ITS groups, only consume IDs. ITSs consume DeviceIDs. Table 3 shows the format of IORT Nodes.

Table 3 Node Format

Field	Byte Length	Byte Offset	Description
Generic data:			
Type	1	0	Possible values and their meanings are: • 0: ITS Group. • 1: Named component. • 2: Root complex. • 3: SMMUv1 or SMMUv2. • 4: SMMUv3. • 5: PMCG. • 6: Memory range. • 7-255: Reserved.
Length	2	1	Length of node in bytes.
Revision	1	3	Revision of the IORT node.
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table. This identifier enables other ACPI tables and DSDT objects to locate this node. This field serves in the same capacity as the _UID object associated with ACPI device objects. IMPLEMENTATION NOTE: In the simplest scheme, the Identifier might be set to the node's index in the array of IORT nodes in the parent IORT table. Other schemes are also possible and permitted.
Number of ID mappings	4	8	Number of ID entries in the ID Array.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings. This field has a value of 0 in the case of an ITS that has no IDs.
Data that varies for node type:			
Data specific to a Node	X	16	
ID Section:			
Array of ID mappings	20xN	16+X	ID mapping Array, where N is the number of ID mappings.

ID mappings represent the formula by which an ID from a source is converted to an ID in a destination. For example, for a root complex behind an SMMU, the RID originating from that root complex must be converted to a StreamID in the destination SMMU. With IORT, ID mappings are declared in the source node. Each mapping describes the destination of the IDs, also known as the output, as well as the numerical relationship that must hold between input IDs and output IDs. The format of each entry in the array of ID mappings is shown in Table 4.

Table 4 ID mapping format

Field	Byte Length	Byte Offset	Description
Input base	4	0	The lowest value in the input range.
Number of IDs	4	4	The number of IDs in the range minus one.

Output base	4	8	The lowest value in the output range.
Output Reference	4	12	A reference to the output IORT Node. This field contains the address offset of the IORT Node relative to the start of the IORT. For example, if this ID mapping is for a root complex outputting to an SMMU, the value of this field is the difference between the start of the SMMU IORT node and the start of the IORT.
Flags	4	16	See Table 5.

Table 5 shows the format of the ID flags.

Table 5 ID flags format

Field	Bit Length	Bit Offset	Description
Single mapping	1	0	Single mapping. Apply the output base regardless of the input IDs. This flag is only valid when the mapping is contained inside a named component, root complex node, SMMUv3 or PMCG node.
Reserved	31	1	Reserved, must be zero.

The following sections describe IORT nodes for SMMUs, ITS groups, named components, and root complexes.

3.1.1 IORT node types

The following sections describe each type of IORT node.

3.1.1.1 SMMUv1 or SMMUv2 node

This section describes the format of the IORT node for SMMUv1 or SMMUv2.

Table 6 Node format for SMMUv1 or SMMUv2

Field	Byte Length	Byte Offset	Description
Type	1	0	This field has a value of 3 for SMMUv1 or SMMUv2.
Length	2	1	The length of the node.
Revision	1	3	3
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	The number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
SMMUv1/2 specific data.			
Base address	8	16	The SMMU base address.
Span	8	24	The length of the memory range that is covered by SMMU memory-mapped IO.

Model	4	32	Possible values are: • 0: Generic SMMUv1. • 1: Generic SMMUv2. • 2: Arm Corelink™ MMU-400. • 3: Arm Corelink™ MMU-500. • 4: Arm Corelink™ MMU-401. • 5: Cavium ThunderX SMMUv2. • All other values are reserved.
Flags	4	36	The SMMU flags. See Table 7.
Reference to Global Interrupt Array	4	40	The offset from the start of this IORT node to the start of its global interrupt array section.
Number of context interrupts	4	44	The number of context interrupts.
Reference to Context Interrupt Array	4	48	The offset from the start of this IORT node to the start of its context interrupt array section.
Number of PMU interrupts	4	52	The number of PMU Interrupts.
Reference to PMU Interrupt Array	4	56	The offset from the start of this IORT node to the start of its PMU interrupt array section.
Global Interrupt Array section			
SMMU_NSgIrpt	4	--	The GSIV of the SMMU_NSgIrpt interrupt.
SMMU_NSgIrpt interrupt flags	4	--	The SMMU_NSgIrpt interrupt flags. See Table 8.
SMMU_NSgCfgIrpt	4	--	The GSIV of the SMMU_NSgCfgIrpt interrupt. This field has a value of 0 if not implemented.
SMMU_NSgCfgIrpt interrupt flags	4	--	The SMMU_NSgCfgIrpt interrupt flags. See Table 8.
Context Interrupts Array section			
Context Interrupts Array	8xN	--	Each interrupt is described by two 4 byte fields: Bytes 0:3: GSIV of interrupt. Bytes 4:7: Interrupt flags as described in Table 8. For SMMUv2 implementations, there must be exactly one interrupt per context bank. In the case of a single, combined interrupt, it must be listed multiple times. Interrupts are indexed by context bank number.
PMU Interrupt Array section			
PMU Interrupt Array	8xN	--	Each interrupt is described by two 4 byte fields: Bytes 0:3: GSIV of interrupt. Bytes 4:7: Interrupt flags as described in Table 8. Interrupts must be ordered by PMU group. That is, for every implemented PMU group an Interrupt entry must be provided. The order of the entries reflects the order of the PMU groups.
IDs for SMMUv1/2 section			
Array of ID mappings	20xN	--	ID Array. N is the Number of ID mappings.

Table 7 describes the SMMUv1 and SMMUv2 table flags.

Table 7 SMMUv1 and SMMUv2 table flags

Field	Bit Length	Bit Offset	Description
DVM Supported	1	0	1: The SMMU supports *Distributed Virtual Memory* (DVM) messages and therefore supports broadcast TLB maintenance operations. 0: The SMMU does not support DVM.
Coherent Page Table Walk	1	1	1: The page table walk done by the SMMU is coherent with CPU caches. 0: The page table walk done by the SMMU is not coherent with CPU caches.
Reserved	30	2	Reserved, must be zero.

Table 8 describes the format of the interrupt flags.

Table 8 Interrupt flags

Field	Bit offset	Number of bits	Description
Edge or level	0	1	1: The interrupt is edge-triggered. 0: The interrupt is level-triggered.
Reserved	1	31	Must be zero.

ID mappings that are defined in an SMMU IORT node can only have an ITS Group node as an output reference, or no IDs in the case of a system that does not have ITS units. All other object types are explicitly forbidden. By implication, ID mappings of an SMMU cannot have an SMMU as an output. In other words, nesting of SMMUs is not allowed.

3.1.1.2 SMMUv3 node

Table 9 shows the format of an SMMUv3 IORT node.

Table 9 SMMUv3 Format

Field	Byte Length	Byte Offset	Description
Type	1	0	4 – SMMUv3.
Length	2	1	Length of node.
Revision	1	3	4
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	Number of ID mappings.
Reference to ID Array	4	12	Offset from the start of an IORT node to the start of the following ID array section.
SMMUv3 specific data			

IO Remapping Table

Field	Byte Length	Byte Offset	Description
Base address	8	16	Base address of SMMU.
Flags	4	24	See Table 10.
Reserved	4	28	Reserved must be zero.
VATOS address	8	32	Optional, set to zero if not supported.
Model	4	40	Possible values are: • 0: Generic SMMU-v3. • 1: HiSilicon Hi161x SMMU-v3. • 2: Cavium CN99xx SMMU-v3. • All other values are reserved for future use.
Event	4	44	GSIV of the Event interrupt if SPI based. Otherwise set to zero.
PRI	4	48	GSIV of the PRI interrupt if SPI based. Otherwise set to zero.
GERR	4	52	GSIV of the GERR interrupt if GSIV based. Otherwise set to zero.
Sync	4	56	GSIV of the Sync interrupt if GSIV based. Otherwise set to zero.
Proximity domain	4	60	If the Proximity Domain Valid flag (see Table 10) is set to 1, this entry provides the proximity domain to which this SMMU instance belongs. An operating system might use this information to optimize performance of the SMMU in a NUMA system. For more information, see SRAT table in [ACPI].
DeviceID mapping index	4	64	If all the SMMU control interrupts are GSIV based, this field is ignored. Where the SMMU uses message signaled interrupts for its control interrupts, this entry contains an index into the array of ID mapping. The indexed ID entry describes the DeviceID and must: • Have the single mapping flag set. Regardless of the flags, the OSPM must ignore the input base and length fields. • Have an output reference that points to an ITS group.
IDs for SMMUv3			
Array of ID mappings	20xN	68	ID Array. N is the number of ID mappings.

Table 10 shows the SMMUv3 flags. When using wired interrupts, the SMMU architecture requires them to be edge sensitive, and therefore no interrupt flags are described in the SMMUv3 IORT node.

ID mappings of an SMMUv3 node can only have ITS group nodes as output references. All other output references are illegal and forbidden.

Table 10 SMMUv3 flags

Field	Bit offset	Number of bits	Description
COHACC Override	0	1	Overrides the value in SMMU_IDR0.COHACC. The OS must use this value in this field if it differs from SMMU_IDR0.COHACC.
HTTU Override	1	2	Overrides the value in SMMU_IDR0.HTTU. The OS must use this value in this field if it differs from SMMU_IDR0.HTTU.
Proximity Domain Valid	3	1	Set to 1 if the value provided in the Proximity Domain field is valid. Set to 0 otherwise.
Reserved	4	28	Must be zero.

3.1.1.1 Performance Monitoring Counter Group node

System components and SMMUs that are based on the SMMUv3 archictecture, can contain Performance Monitoring Counter Groups (PMCG). A PMCG node contains a reference to an SMMU, named component, or a root complex node that is associated with that PMCG instance. Note that the association of PMCGs to components is implementation defined and therefore system specific knowledge is required to use them.
Table 11 describes the PMCG node:

Table 11 Performance monitoring counter group

Field	Byte Length	Byte Offset	Description
Type	1	0	5 – PMCG.
Length	2	1	The length of the node.
Revision	1	3	2
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	This field can be zero or one. If it is zero it indicates that the overflow interrupt is wire based, and GSIV field below describes the vector. If it is one, a single ID mapping describes the Device ID and ITS group for the PMCG interrupt.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings. The field is set to zero if there is no ID mapping.
PMCG specific data:			
Page 0 base address	8	16	Page 0 base address for performance monitor counter group.
Overflow interrupt GSIV	4	24	GSIV of overflow interrupt if the interrupt is wire based, set to zero otherwise.
Node reference	4	28	Offset from the start of the IORT table to the start of the SMMUv3, Root complex, or Named component node that is associated with this PMCG.

Page 1 base address	8	32	Page 1 base address for performance monitoring counter group. This field is ignored by the OS if SMMU_PMCG_CFGR. RELOC_CTRS is zero.
IDs for Named component:			
ID mapping	20	--	ID Mapping if Number of IDs is one. This field may be omitted.

PMCG group structures can describe the PMCG interrupt as a wired interrupt or MSI. For the latter case the structures can provide an ID mapping, encompassing the device ID and target ITS group of the MSI. In this case the mapping should have the single mapping flag set. Regardless of the flags, the OSPM must ignore the input mapping base and length fields. When using wired interrupts, the SMMU architecture requires them to be edge sensitive, therefore no interrupts flags are described in the table.

3.1.1.2 ITS group node

ITS group nodes describe which ITS units are in the system. A node allows grouping of more than one ITS, but all ITSs in the group must share a common understanding of DeviceID values. That is, a given DeviceID must represent the same device for all ITS units in the group.

ITS group nodes have no ID mappings. Table 12 shows the format of ITS groups.

Table 12 ITS Group Format

Field	Byte Length	Byte Offset	Description
Type	1	0	This field has a value of 0.
Length	2	1	The length of the node in bytes.
Revision	1	3	1
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	This field has a value of 0. ITS groups do not have IDs.
Reference to ID Array	4	12	This field has a value of 0. There is no ID array.
ITS specific data			
Number of ITSs	4	16	The number of ITSs.
GIC ITS Identifier Array	4xN	20	The array of ITS identifiers. These IDs must match the value used in the *Multiple APIC Description Table* (MADT) GIC ITS structure for each relevant ITS unit. See [ACPI].

3.1.1.3 Named component node

Named component nodes are used to describe devices that are also included in the *Differentiated System Description Table* (DSDT). See [ACPI].

These nodes can have one or more ID mappings, and the mappings can use SMMUs or ITS groups as output references. All other output node types are forbidden as output references.

It is permitted for these nodes to declare and use implementation-defined input IDs. Such input IDs, if present, are considered to be specific to the Named Component only, and their interpretation is entirely relegated to the device driver that is managing the Named Component.

It is to be noted that if implementation-defined input IDs are used in an ID mapping which has the single mapping flag set, then OSPM must ignore the input base and length fields.

Table 13 describes the IORT node format for Named components.

Table 13 Named component node format

Field	Byte Length	Byte Offset	Description
Type	1	0	For a named component, this field has a value of 1.
Length	2	1	The length of the node.
Revision	1	3	4
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	The number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
Named component specific data:			
Node flags	4	16	Bits 31:6 Reserved, must be zero Bits 5:1 Substream width Number of substream bits supported by this device, the value n indicates support for substream values 0 through 2^n-1. A value of 0 indicates no substream support. Bit 0: Stall supported • This bit has a value of 1 if the system can tolerate transactions being stalled for this named device. • This bit has a value 0 if the system cannot tolerate transactions being stalled for this named device. WARNING: incorrectly setting this bit can lead to system deadlock and instability. If in doubt, set the bit to zero.
Memory access properties	8	20	These properties are described in Table 14.
Device memory address size limit	1	28	The number of address bits, starting from the least significant bit that can be used by a device when it accesses memory. NOTE: If the named component or a bus upstream of the device includes a _DMA method, then the OS must take into account the memory access limits returned by the method as well as the value in this field, to determine the overall memory access range for this named component. The most restrictive of the two values must be chosen as the limit for memory accesses issued by this named component.
Device object name	--	29	The ASCII Null terminated string with the full path to the entry in the namespace for this object.
Padding	--	--	Padding is to 32-bit word-aligned.

IO Remapping Table

Field	Byte Length	Byte Offset	Description
IDs for Named component:			
Array of ID mappings	20xN	--	ID Array. N is the Number of ID mappings.

Table 14 describes device node memory access properties.

Table 14 Memory access properties

Field	Byte Length	Byte Offset	Description
CCA: Cache Coherent Attribute	4	0	This value must match the value returned by the _CCA object that is defined in the DSDT for the device represented by this node. The attribute can take the following values: 0x1: The device is fully coherent. No cache maintenance* is required for memory that is shared with the device that is mapped on CPUs as *Inner Write-Back* (IWB)*, Outer Write-back* (OWB)*, and Inner shareable* (ISH). In addition, during system initialization at cold boot, or after wakeup from low-power state, if the cache coherency requires an SMMU override or some specific device configuration, the platform firmware has to ensure that this has been done. Therefore the semantics represented by a value of 0x1 are always correct at the time of hand-off from firmware to OS. 0x0: The device is not coherent. Therefore: • Cache maintenance is required for memory that is shared with the device that is mapped on CPUs as IWB-OWB-ISH. • No cache maintenance is required for memory that is shared with the device that is mapped on CPUs as device or Non-cacheable. All other values are reserved.
AH: Allocation Hints	1	4	This field can be ignored without loss of correctness. Allocation hints have the following format: • Bits[7:4] are ignored by the OS and must be zero. • Bit 3: Allocation Hints Override (AHO). o 0: If the bit is clear, use the incoming Read Allocate (RA), Write Allocate (WA), and Transient (TR) hints. o 1: If the bit is set, override allocation hints based on the values in bits RA, WA, and TR. • Bit 2: Read Allocate (RA). o 0: Clear read allocation hint if AHO is set to 1. o 1: Set read allocation hint if AHO is set to 1. • Bit 1: Write Allocate (WA). o 0: Clear write allocation hint if AHO is set to 1. o 1: Set write allocation hint if AHO is set to 1. • Bit 0: Transient (TR). o 0: Clear transient hint if AHO is set to 1.

Field	Byte Length	Byte Offset	Description
			○ 1: Set transient hint if AHO is set to 1.
Reserved	2	5	Reserved, must be zero.
MAF: Memory Access Flags	1	7	See Table 15.

* Note: Caching operations described in this document apply to the CPU caches and any other caches in the system where device memory accesses can hit.

Table 15 describes the memory access properties flags field format.

Table 15 Memory Access Flags

Field	Bit Length	Bit Offset	Description
CPM: Coherent Path to Memory	1	0	0x1: If set, this indicates that the device has path to memory that allows coherency with the CPU cache hierarchy. This means that if the CPU maps the memory as IWB, OWB, ISH, and the device is outputting the same attributes, or being overridden through an SMMU to provide the same attributes, no cache maintenance is required. 0x0: The device does not have a path to memory that is coherent with the CPU cache hierarchy. Therefore: • Cache maintenance is required for memory that is shared with the device that is mapped on the CPU as IWB-OWB-ISH. • No cache maintenance is required for memory that is shared with the device that is mapped on the CPU as device or Non-cacheable. Note that if CCA is 0x1, CPM must also be 0x1. Conversely, If CPM is 0x0 then CCA must be 0x0. However, it possible for the system to boot with CCA set to 0x0 and CPM set to 0x1. See Table 16 for further details.
DACS: Device attributes are Cacheable and Inner-Shareable	1	1	The device outputs IWB-OWB-ISH attributes: 0x1: The device outputs IWB-OWB-ISH attributes. 0x0: The device does not output IWB-OWB-ISH attributes.
Reserved	6	2	Reserved must be zero.

Not every combination of memory attribute values is valid or useful. Table 16 lists the valid combinations and their uses:

Table 16 Valid Memory Attributes

CCA	CPM	DACS	AH	Comment
1	1	1	Can be applied	The device outputs the correct attributes that enable cache coherency (IWB-OWB-ISH). If the shared memory is mapped on the CPU with the same attributes, no cache management is necessary, and the device exploits cache coherency.

CCA	CPM	DACS	AH	Comment
				If the device is behind an SMMU, the OS might override the allocation hint attributes using the values that are supplied in the AH. However, in this case the OS must maintain the coherency guarantee indicated by the CCA value of 1. Therefore, it must not change the cacheability and shareability attributes provided by the device (IWB-OWB-ISH).
1	1	0	Can be applied	The device does not natively provide IWB-OWB-ISH attributes, and cache coherency is provided by an override through an SMMU. The value of 1 for CCA shows that boot firmware has configured the SMMU to ensure cache coherency. The OS might apply the allocation hints supplied in AH when overriding the device attributes. A device with this combination of flag values must be behind an SMMU. Therefore, it must have an ID in its array of ID mappings that has an SMMU IORT node as its output reference.
1	0	-	N/A	Illegal. If CPM is 0, CCA cannot be 1.
0	1	0	Can be applied	The device does not natively provide IWB-OWB-ISH attributes, but cache coherency can be provided by an override through an SMMU. The OS can provide cache coherency by using an SMMU to override the device attributes to IWB-OWB-ISH. In this case, the OS might apply the allocation hints supplied in AH when overriding the device attributes. A device with this combination of flag values must be behind an SMMU. Therefore, it must have an ID in its array of ID mappings that has an SMMU IORT node as its output reference.
0	1	1	Can be applied	Illegal. If the device has a coherent path to memory, and natively outputs IWB-OWB-ISH attributes then CCA must be set to 1.
0	0	*	N/A	The device is not coherent and cannot be made coherent. If the CPU maps memory that is shared with the device as IWB-OWB-ISH, cache maintenance is required. If the CPU maps the memory as Non-cacheable, then no cache maintenance is required.

Appendix B describes how an OS can use this information and when caching operations are required.

3.1.1.4 PCI root complex node

This specification uses the term root complex, to refer to a PCIe domain that consists of a set of PCIe devices interconnected in a PCIe hierarchy, where:

- The domain is wired to a common host SMMU or ITS group.
- The PCIe devices belonging to the domain share common cacheability and memory accessibility properties.

The root complex node is used to describe a root complex. This is a logical root complex in software, and need not be the same as a physical root complex implementation. The root complex node can thus be used to represent any of the following as long as they meet the requirements of a PCIe domain:

- An RCiEP or a set of RCiEPs
- A standalone PCIe device
- A PCIe root port or a set of PCIe root ports
- A PCIe host bridge that has a set of PCIe devices below it, including PCIe root ports and RCiEPs
- A physical root complex with multiple host bridges and PCIe devices below it

The format of the root complex node is described in Table 17. In addition, the following rules and assumptions apply:

- ID mappings can use SMMUs or ITS groups as output references.
- A range of Requester IDs must be mapped to one and only one SMMU.
- If there are multiple root complex nodes with the same PCIe segment number, then their Requester ID ranges must not overlap. For such root complex nodes, the firmware must include the PCI Firmware defined _DSM for ignoring PCI boot configuration, Function 5, in the ACPI device object of each PCIe host bridge in ACPI namespace that belongs to the 256-bus PCIe segment. The _DSM method should return a value of 0 to indicate that the OS must preserve the PCI configuration that the firmware has performed at boot time. See [PCIFW] for more details on this _DSM method.

Table 17 Root Complex Node

Field	Byte Length	Byte Offset	Description
Type	1	0	For the root complex type, this field has a value of 2.
Length	2	1	The length of the node.
Revision	1	3	4
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	The number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
Root complex specific data.			
Memory access properties	8	16	These properties are described in Table 14. Note that SBSA [SBSA] requires root complexes to support IO coherency and to be in the same shareability domain as the CPUs.
ATS Attribute	4	24	Support for ATS and its ancillary features. ATS must be mandatorily enabled for any ancillary feature to also be enabled. Bit 2: 1: The root complex supports forwarding of PASID information on translated transactions to the SMMU 0: The root complex does not support forwarding of PASID information on translated transactions to the SMMU Bit 1: 1: The root complex supports PRI

			0: The root complex does not support PRI Bit 0: 1: The root complex supports ATS 0: The root complex does not support ATS Bits [31:3] are reserved and must be zero.
PCI Segment number	4	28	The PCI segment number, as in MCFG and as returned by _SEG in the namespace.
Memory address size limit	1	32	The number of address bits, starting from the least significant bit that can be used by the root complex when it accesses memory. NOTE: If a _DMA method is present upstream of this root complex in its PCIe hierarchy, then the OS must take into account the memory access limits returned by the method as well as the value returned in this field, to determine the overall memory access limits for this device. The most restrictive of the two values must be chosen as the limit for memory accesses issued by this root complex.
PASID capabilities	2	33	Bits [4:0]: Max PASID Width Indicates the width of the PASID field supported by the root complex. The value n indicates support for PASID values 0 through 2n-1 (inclusive). The value 0 indicates support for a single PASID (0). The value 20 indicates support for all PASID values (20 bits). This field must be between 0 and 20 (inclusive). Bits [15:5] are reserved and must be zero.
Reserved	1	35	Reserved must be zero.
Flags	4	36	Bit 0: PASID support 1: The root complex supports PASID. 0: The root complex does not support PASID. Other bits are reserved for future use and must be set to zero.
IDs for root complex			
Array of ID mappings	20xN	--	Array of IDs. N is the Number of ID mappings.

It is expected that all accesses from PCIe devices behind the root complex are cache coherent and in the same inner shareability domain as the CPUs controlled by the OS that is parsing these tables.

3.1.1.5 Reserved Memory Range node

The Reserved Memory Range (RMR) node is used to describe memory ranges that are reserved for use by endpoints. Reserved memory ranges require a unity mapping in the SMMU.

A memory range can be reserved for dedicated use by a set of requesters. Each Requester ID (RID) in the set is mapped to a unique corresponding Stream ID that is input to an SMMU in the system. The RMR node provides a mapping of the form {set of reserved memory ranges, associated Stream IDs}. The association of reserved memory ranges to Stream IDs is always expressed as *M:N* for each RMR node, where *M* is the number of reserved memory ranges specified by that node, and *N* is the number of Stream IDs that the *M* memory ranges are reserved for. The output Stream IDs are expressed through the ID mappings.

IO Remapping Table

The RMR is depicted in Figure 1.

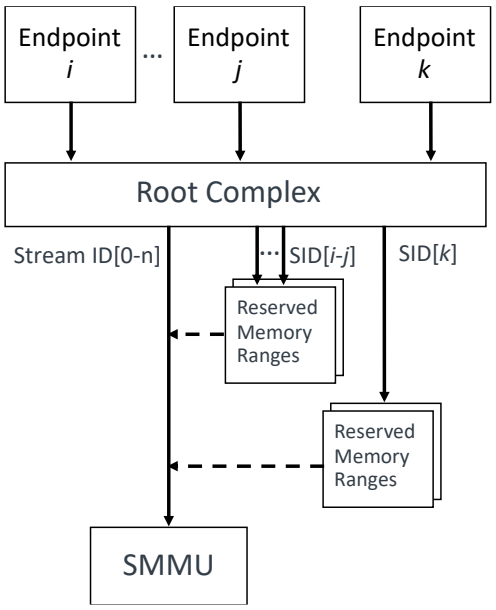

Figure 1: Association of Reserved Memory Ranges with Stream IDs

Table 18 Reserved Memory Range (RMR) node

Field	Byte Length	Byte Offset	Description
Type	1	0	For the Reserved Memory Range type, this field has a value of 6.
Length	2	1	The length of the node.
Revision	1	3	3
Identifier	4	4	Unique identifier for this node that can be used to locate it in the parent table.
Number of ID mappings	4	8	Number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
RMR specific data:			
Flags	4	16	Flags that apply to this node. See Table 20 for details.
Number of Memory Range Descriptors	4	20	Number of Memory Range Descriptors.
Reference to Memory Range Descriptors	4	24	Offset from the start of the IORT node to the start of its Array of Memory Range descriptors. The Memory Range Descriptor is defined in Table 19.

IO Remapping Table

Table 19 describes the format of the Memory Range Descriptor.

Table 19 Memory Range Descriptor

Field	Byte Length	Byte Offset	Description
Physical Range offset	8	0	Base address of the Reserved memory Range, aligned to a page size of 64K.
Physical Range length	8	8	Length of the Reserved Memory range. Must be a multiple of the page size of 64K.
Reserved	4	16	Reserved, must be zero.

Table 20 describes the Node flags field.

Table 20: RMR node flags

Field	Bit Length	Bit Offset	Description
Remapping Permitted	1	0	0x1: Allow OS to remap reserved memory ranges. 0x0: Disallow remapping of reserved memory ranges.
Access privilege	1	1	0x1: Requester accesses to the memory ranges of this node are marked as privileged. 0x0: Requester accesses to the memory ranges of this node are marked as unprivileged.
Access attributes	8	2	Attributes that apply to accesses to the memory ranges described by this node: 0x00: Device-nGnRnE memory 0x01: Device-nGnRE memory 0x02: Device-nGRE memory 0x03: Device-GRE memory 0x04: Normal Inner Non-cacheable Outer Non-cacheable 0x05: No rmal Inner Write-back Outer Write-back Inner Shareable Other values are reserved for future use by this specification.
Reserved	22	10	Reserved, must be zero.

Each Memory Range descriptor in an RMR node must describe a unique range of memory that does not overlap with memory ranges described by other descriptors within the node.

The attributes of an RMR node, such as access privileges, apply to all memory ranges described by the RMR node. If a requester or set of requesters requires memory ranges with mutually dissimilar attributes, then it must use multiple RMR nodes, one RMR node for each memory range. It is permitted to describe memory ranges with identical attributes that are reserved by a set of Stream IDs using a single RMR node. This is called the RMR node compaction.

As explained earlier, the ID mapping of the memory range node maps the memory ranges defined by the Memory Range Descriptor in Table 19 to one or more output Stream IDs. The input base is ignored. The Number of IDs

field must be valid and should apply to the output range, and reflects the number of output Stream IDs associated with the memory ranges that are described by this node.

If reserved memory regions are present, the OS must preserve PCIe configuration performed by the boot firmware. This preservation is required to ensure functional continuity of the endpoints that are using the reserved memory regions. Therefore, RMR nodes must be supported by the inclusion of the PCI Firmware defined _DSM for ignoring PCI boot configuration, Function 5, in the ACPI device object of the PCIe host bridge in ACPI namespace. The _DSM method should return a value of 0 to indicate that the OS must honour the PCI configuration that the firmware has done at boot time. See [PCIFW] for more details on this _DSM method.

NOTE: Arm strongly discourages the use of reserved memory ranges. Reserved memory ranges create holes in the IOVA space that can cause loss of flexibility for the operating system in terms of its ability to remap memory ranges. Furthermore, in virtualized systems, this additionally imposes constraints on device reassignments to Virtual Machines.

Appendix A Rationale and Examples

Figure 2 depicts an example of a system with a relatively complex topology.

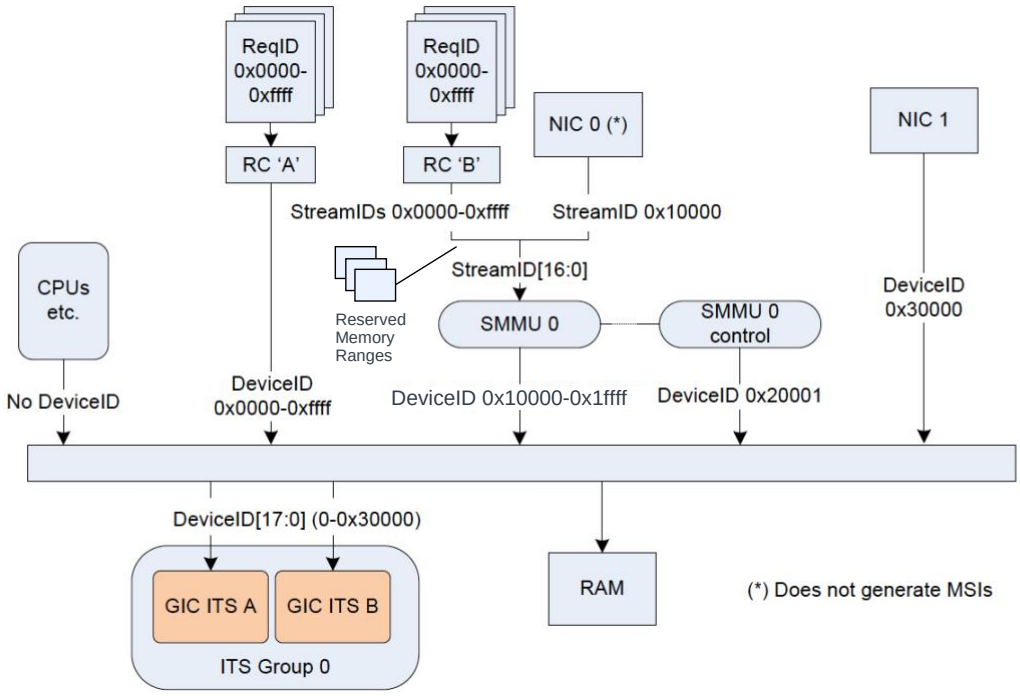

Figure 2 Example system

The example system is depicted as comprising a set of components that can communicate with a group of GIC ITSs and devices behind an SMMU. Figure 3 provides a different view of the example system. It depicts the IDs for each component across the various ID spaces, RID, StreamID, and DeviceID. It also depicts a set of memory ranges for which unity mapping is required.

The first obvious feature of the system is that there are two overall types of ID that can be described for a given component:

- Some component, such as root complexes, require a range of IDs.
- Some components, for example NIC1 or NIC0, require only a single ID.

Two kinds of ID definitions emerge, a range and an endpoint.

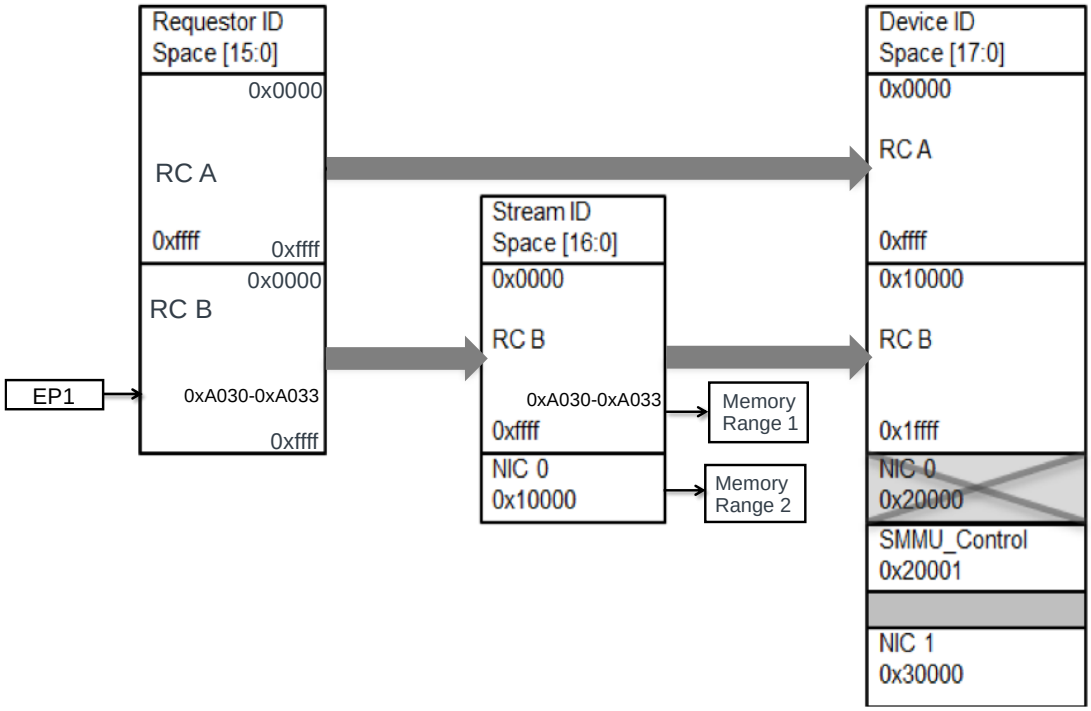

Figure 3 ID mappings

Figure 3 also shows how IDs in one space map to another. For example, the RID space of RC B, which spans 0x0->0xffff, maps to StreamID space 0x0000->0xffff in SMMU 0, which in turn maps to DeviceID space 0x10000->0x1ffff in the ITS group. This can be viewed as an input ID to output ID mappings, for example RC B has an input ID in the RID space and an output ID in the StreamID space of SMMU 0. However, this rule does not apply to every device, because some devices, such as the endpoints NIC 0 and NIC 1 in the example, only have an output ID. Finally, not every device is capable of generating MSIs, so some devices, such as NIC 0 in the example system, can only have a StreamID mapping. The mappings in Figure 3 can be described by the following pseudocode:

```
RC A
      // doesn't use SMMU 0 so just outputs DeviceIDs to ITS GROUP 0
      // Input ID    --> Output reference: Output ID
      0x0000-0xffff  --> ITS GROUP 0 : 0x0000->0xffff
RC B
      // Input ID   --> Output reference: Output ID
      0x0000-0xffff --> SMMU 0    : 0x0000->0xffff
NIC 0
      // endpoint device doesn't have an explicit input ID
      // In this example the NIC 0 does not generate MSI so
      // has no DeviceID
```

IO Remapping Table

```
      // Input ID  --> Output reference: Output ID
      N/A          --> SMMU 0     : 0x10000
SMMU 0
      // Note that range of StreamIDs that map to DeviceIDs excludes
      // the NIC 0 DeviceID as it does not generate MSIs
      // Input ID   --> Output reference: Output ID
      0x0000-0xffff --> ITS GROUP 0 : 0x10000->0x1ffff
      // SMMU 0 Control interrupt is MSI based
      // Input ID   --> Output reference: Output ID
      N/A           --> ITS GROUP 0 : 0x200001
NIC 1
      // end point device doesn't have an explicit input ID
      // Input ID   --> Output reference: Output ID
      N/A           --> ITS GROUP 0 : 0x30000
```

In the example above, each component declares any IDs it owns. For each ID, it provides detail of how that ID maps to the ID of another component. The pseudocode describes both the topological arrangement of the system and the ID relationships. For IDs that cover a range, the IDs described allow computing input to output mappings by using the following simple offsetting:

OutputID = ID – Input Base + Output base

For example, RC B has an ID range that can be described as follows:

0x0000-0xffff --> SMMU 0 : 0x0000->0xffff

That is, RID range 0x0000-0xffff maps to the StreamID range of 0x0000-0xffff belonging to SMMU 0. Therefore, for example, RID 0x0003, when emitted from RC B, maps to a StreamID as follows:

StreamID 0x0003 = RID 0x0003 – RID base 0x0000 + StreamID base 0x0000

In the example, the DeviceID mapping follows from the ID mapping for SMMU 0, which has the following format:

0x0000-0xffff --> ITS GROUP 0 : 0x10000->0x1ffff

The StreamID range 0x0000 to 0xffff for SMMU 0 maps to the DeviceID range of 0x10000 to 0x1ffff for ITS GROUP 0. Therefore, the StreamID of 0x0003 maps as follows:

DeviceID 0x10003 = StreamID 0x0003 - StreamID base 0x0000 + DeviceID base 0x10000

Endpoint IDs are described directly. For example, NIC 1 has a DeviceID of 0x30000. Effectively, these IDs can be treated as range mappings, where the range contains only one ID.

Note that the resolution of the ID space for a PCIe root complex, an SMMU, or an ITS might not always increase monotonically, as one moves further out in the system, that is, from PCIe RID (16 bits) to SMMU (IMPLEMENTATION DEFINED) or ITS (IMPLEMENTATION DEFINED). For example, an SMMU might implement an 8-bit StreamID space. The hardware can be arranged so that not every bit of the RID is emitted to the SMMU. For example, the following relationship might hold:

- StreamID bits[5:0] = RID bits[5:0].

- StreamID bits[7:6] = RID bits[9:8].

This type of relationship can still be described in pseudocode by describing the individual mappings that arise:

```
RC X
      // Input ID      --> Output reference: Output ID
      0x0000-0x003F --> SMMU Y    : 0x00->0x3F
      // 0x0040-0x00FF // Invalid range
      0x0100-0x013F --> SMMU Y    : 0x40->0x7F
      // 0x0140-0x01FF // Invalid range
      0x0200-0x023F --> SMMU Y    : 0x80->0xBF
      // 0x0240-0x02FF // Invalid range
      0x0300-0x033F --> SMMU Y    : 0xC0->0xFF
      // 0x0340-0xFFFF // Invalid range
```

All of the relationships described in this example can be described using the ID mapping arrays provided by the IORT nodes. See Table 3 and Table 4 for further detail. The following sections provide some examples.

Representing components that generate MSIs and are connected to an SMMU

In the example system, root complex B is connected to SMMU 0 which in turn connects to ITS GROUP 0. The ID relationships for a PCIe device behind root complex B are as follows:

- StreamID for SMMU 0 = RID.
- DeviceID for ITS GROUP 0 = StreamID + 0x10000.

The resulting IORT Node for root complex B has the following single ID entry:

Field	Description
Number of IDs	0xffff
Flags	0x0
Input Base	0x0
Output base	0x0
Output Reference	Offset from the start of the IORT to the start of the SMMU 0 node.

The ID table maps the whole 16-bit RID range, bit by bit, to a 16-bit StreamID range.
The node for SMMU 0 has the following ID entry:

Field	Description
Number of IDs	0xffff
Flags	0x0
Input Base	0x0
Output base	0x10000

IO Remapping Table

Field	Description
Output Reference	Offset from start of the IORT to the start of the ITS GROUP 0 node.

The ID table maps the 16-bit StreamID range to a DeviceID range, with an offset of 0x10000.

Representing components that generate MSIs but are not connected to an SMMU

In the example, root complex A, RC A, can generate MSIs but is not connected to an SMMU. For RC A the IORT Node ID array has the following single entry:

Field	Description
Number of IDs	`0xffff`
Flags	`0x0`
Input Base	`0x0`
Output base	`0x0`
Output Reference	Offset from the start of the IORT to the start of ITS GROUP 0.

This example maps one-to-one between a RID and a DeviceID, that is, DeviceID = RID.

NIC 1 also follows this pattern, and has the following ID mapping in its IORT node:

Field	Description
Number of IDs	`0x0`
Flags	`0x0`
Input Base	`0x0`
Output base	`0x30000`
Output Reference	Offset from the start of the IORT to the start of ITS GROUP 0.

This mapping results in a DeviceID of 0x30000 for NIC 1.

Representing Components that do not generate MSIs but are connected to an SMMU

There are essentially two options in this category of components:

1. Mappings are provided that generate a DeviceID for the device, but as the device driver does not use MSIs, the DeviceID remains unused.
2. Mappings are provided that make it impossible to reserve a DeviceID for the device.

Option 2 is described below and applies to NIC 0 in the example system. This device represents its StreamID relationship to SMMU 0 with the following ID entry:

Field	Description
Number of IDs	0x0
Flags	0x0
Input Base	0x0
Output base	0x10000
Output Reference	Offset from the start of the IORT to the start of the SMMU 0 node.

StreamID 0x10000 is reserved in SMMU 0 for NIC 0. For SMMU 0, the following ID mapping is declared for StreamID to DeviceID conversion:

Field	Description
Number of IDs	0xffff
Flags	0x0
Input Base	0x0
Output base	0x10000
Output Reference	Offset from the start of the IORT to the start of an ITS GROUP 0.

The omission of a StreamID to DeviceID mapping for StreamID 0x10000, generated by NIC 0, implies that NIC 0 does not generate MSIs.

Representing reserved memory ranges

Reserved memory ranges are expressed within a memory range node.

For endpoint EP1 in Figure 3, the Root Complex node for RC B first maps Requester ID from EP1 to a corresponding output Stream ID as follows:

```
RC B
      // Input ID      --> Output reference: Output ID
      0x0000-0xFFFF --> SMMU 0    : 0x0000->0xFFFF
```

According to this mapping equation, Requester IDs belonging to EP1 are automatically mapped to the Stream ID range 0xA030-0xA033.

A memory range node for Memory Range 1 is then defined as follows:

IO Remapping Table

Field	Byte Length	Byte Offset	Description
Type	1	0	0x6
…	…	…	…
Number of ID mappings	4	8	1
Reference to ID Array	4	12	ID mapping of this reserved memory range node.
Number of Memory Range Descriptors	4	16	1
Memory Range Descriptors [1]	-	20	Memory range descriptor for Memory Range 1.

The reserved memory range node includes a single memory range descriptor of the following format:

Field	Byte Length	Byte Offset	Description
Physical Range offset	8	0	Base address of Memory Range 1.
Physical Range length	8	8	Size of Memory Range 1.
Reserved	4	16	0

The reserved memory range node describes a single ID mapping entry as follows:

Field	Description
Number of IDs	0x4
Input Base	0 (Ignored)
Output base	0xA030
Output Reference	Offset from the start of the IORT to the start of SMMU 0 node.

In this mapping, the Number of IDs field indicates that the reserved memory ranges described by this RMR node are associated with four Stream IDs starting with 0xA030.

A separate memory range node must be defined for NIC 0, as follows:

Field	Byte Length	Byte Offset	Description
Type	1	0	0x6
…	…	…	…
Number of ID mappings	4	8	1
Reference to ID Array	4	12	ID mapping of this reserved memory range node.

Number of Memory Range Descriptors	4	16	1
Memory Range Descriptors [1]	-	20	Memory range descriptor for Memory Range 2.

The reserved memory range node includes a single memory range descriptor of the following format:

Field	Byte Length	Byte Offset	Description
Physical Range offset	8	0	Base address of Memory Range 2.
Physical Range length	8	8	Size of Memory Range 2.
Reserved	4	16	0

The reserved memory range node describes a single ID mapping entry as follows:

Field	Description
Number of IDs	0x1
Input Base	0 (Ignored)
Output base	0x10000
Output Reference	Offset from the start of the IORT to the start of SMMU 0 node.

Appendix B OS Usage of Memory Attributes

Figure 4 illustrates how an IORT-aware OS can interpret the memory attribute properties of a device, when the OS maps device shared memory on the CPU as IWB-OWB-ISH.

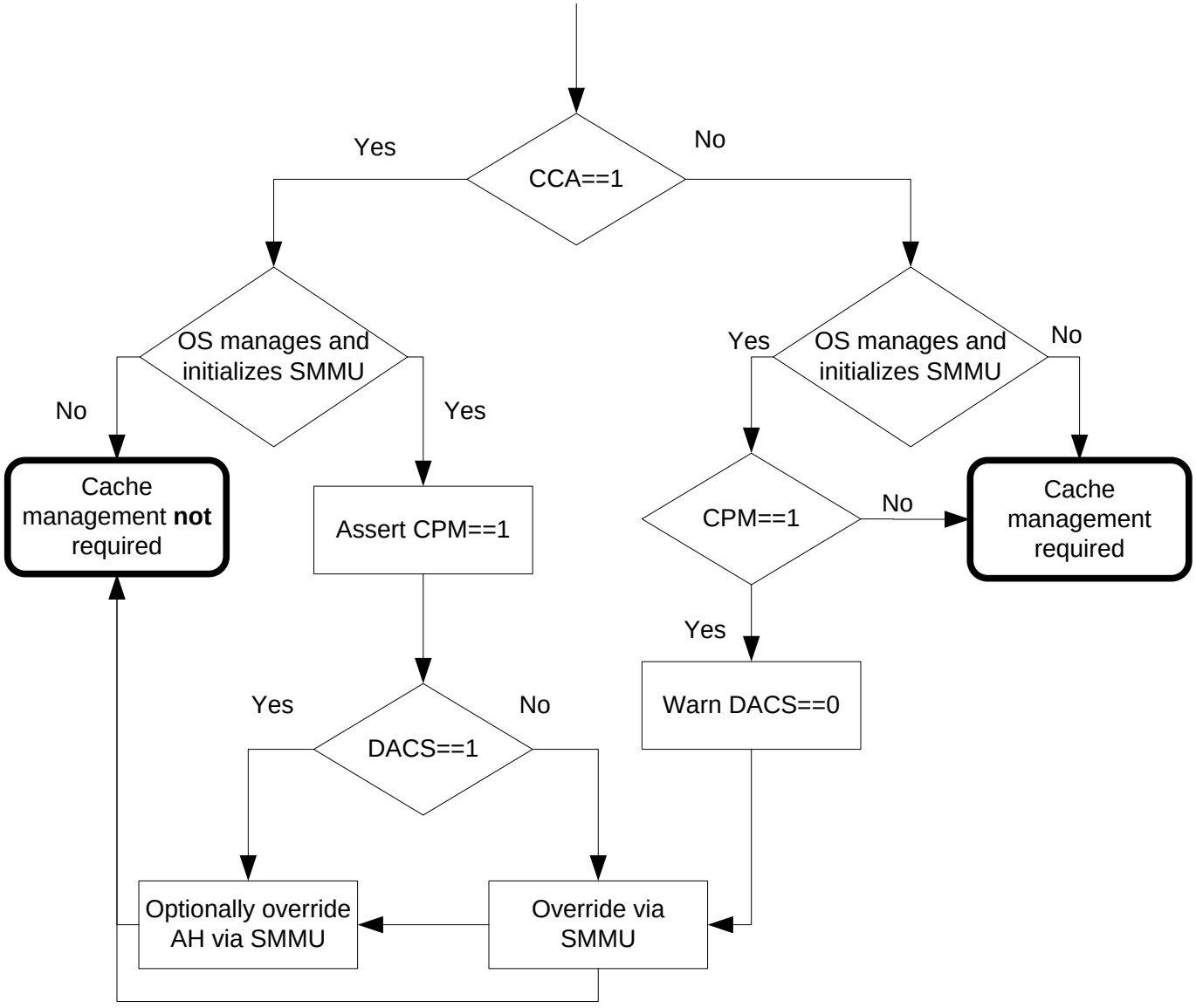

Figure 4 OS decision tree for cache management assuming CPU maps memory as IWB-OWB-ISH

IO Remapping Table

Table 21 describes cache management requirements in more detail, taking into account the values of the flags, the memory mapping type used on the CPU, and the override memory type used by the SMMU, when applicable. In the table, NC represents a Non-cacheable type (Normal non-cacheable or Device).

Table 21 Cache management requirements

CCA	CPM	DACS	SMMU override for memory mapping	CPU memory mapping	Cache Maintenance Required*
1	1	1	None	IWB-OWB-ISH	No
1	1	1	NC	NC	No
1	1	1	NC	IWB-OWB-ISH	Yes
0	1	0	IWB-OWB-ISH	IWB-OWB-ISH	No
0	1	0	None	NC	No
0	1	0	None	IWB-OWB-ISH	Yes
0	0	0	N/A	IWB-OWB-ISH	Yes
0	0	0	N/A	NC	No

* Note: The Caching operations that are described in this document apply to the CPU caches and any other caches in the system where device memory accesses can hit.