Intel® Resource Director Technology (Intel® RDT) Architecture Specification

March 2025

Revision 1.2

Document Number: 356688-003US

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel is a sponsor and member of the Benchmark XPRT Development Community and was the major developer of the XPRT family of benchmarks. Principled Technologies is the publisher of the XPRT family of benchmarks. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

Contents

Figures

Tables

Revision History

Revision Number	Description	Date
1.0	• Initial release of the document.	September 2023
1.1	• Adding details of hardware feature support in future Intel Processors	January 2025
1.2	• Minor clarifications, discussion of Hybrid CPUID enumeration and feature interactions	March 2025

Glossary

Table 1-1 Glossary

Acronym	Term	Description
ACPI	Advanced Configuration and Power Interface	Advanced Configuration and Power Interface is an open standard that operating systems can use to discover and configure computer hardware components, to perform power management, auto configuration, and status monitoring.
CAT	Cache Allocation Technology	Software-guided redistribution of cache capacity is enabled by CAT, enabling important data center VMs, containers or applications to benefit from improved cache capacity and reduced cache contention. CAT may be used to enhance runtime determinism and prioritize important applications.
CDP	Code and Data Prioritization	As a specialized extension of CAT, Code and Data Prioritization (CDP) enables separate control over code and data placement in the L2 cache and the last-level (L3) cache. Certain specialized types of workloads may benefit with increased runtime determinism, enabling greater predictability in application performance.
CH	Channel	An I/O device channel, used to communicate between a device and an I/O Block and onto the coherent fabric.
CLOS	Class(es) of Service	A fundamental tag in RDT used for resource controls
-	Clump	A group of associated register fields within a larger register space (such as a 4KB page)
CMT	Cache Monitoring Technology	Monitors the last-level cache (L3) utilization by individual threads, applications, or Virtual Machines, CMT improves workload characterization, enables advanced resource-aware scheduling decisions, aids “noisy neighbor” detection and improves performance debugging.
-	Hybrid	Term used to refer to processors supporting more than one logical processor type, potentially with differing feature support or attributes details
Intel® RDT	Intel® Resource Director Technology	Intel® RDT is the “umbrella” technology name for Intel’s Platform Quality of Service technologies, including CPU Agents and Non-CPU Agents.
I/O Intel® Resource Director Technology (Intel® RDT)	I/O Device Intel® Resource Director Technology	Intel RDT technologies specifically focusing on I/O devices including PCIe, CXL and integrated accelerators
MBA	Memory Bandwidth Allocation	MBA enables approximate and indirect control over memory bandwidth available to workloads, enabling new levels of interference mitigation and bandwidth shaping for “noisy neighbors” present on the system.

Acronym	Term	Description
MBM	Memory Bandwidth Monitoring	Multiple VMs or applications can be tracked independently via Memory Bandwidth Monitoring (MBM), which provides memory bandwidth monitoring for each running thread simultaneously. Benefits include detection of noisy neighbors, characterization and debugging of performance for bandwidth-sensitive applications, and more effective non-uniform memory access (NUMA)-aware scheduling.
MMIO	Memory Mapped I/O	I/O Intel RDT defines a series of MMIO-mapped interfaces to enable association of I/O devices to RMIDs and CLOS for monitoring and control.
PQR	PQR	A shorthand for the IA32_PQR_ASSOC MSR, which associates IA threads to RMID and CLOS tags.
RMD	Resource Management Domain	A set of features defined within a particular cache domain, such as an L3 cache supporting a number of logical processors.
RTD	Resource Telemetry Domain	A Resource Management Doman within which one or more resource monitoring (telemetry) controls are supported
RAD	Resource Allocation Domain	A Resource Management Doman within which one or more resource allocation controls are supported
RMID	Resource Monitoring ID(s)	A fundamental tag used for resource monitoring in Intel RDT.
SoC or SOC	System-on-Chip	An integrated chip composed of host processors, accelerators, memory, and I/O agents.
TC	Traffic Class	A PCI Express feature that allows differentiation of transactions to apply appropriate servicing policies.
VC	Virtual Channel	A PCI Express feature for differential bandwidth allocation. Virtual channels have dedicated physical resources (buffering, flow control management, and so on) across the hierarchy.
VMM	Virtual Machine Monitor	A software layer that controls virtualization.

1 Introduction

This document defines the architecture of the Intel® Resource Director Technology (Intel® RDT) feature set. The goal of Intel RDT is to bring new levels of monitoring and control over how shared platform resources such as last-level cache (L3) and main memory (typically DRAM) bandwidth are utilized by CPU Agents and non-CPU Agents. The monitoring and allocation are not necessarily applied across the entire system but are applied to a Resource Management Domain (RMD) which corresponds to a set of agents sharing a set of system resources, such as L2 cache capacity, L3 cache capacity, memory bandwidth, and I/O devices. A Resource Management Domain (RMD) consists of a collection of CPU agents or non-CPU agents. The set of CPU agents consist of one or more logical processors associating an RMID and/or CLOS tag with a software thread. Non-CPU agents include PCI Express* (PCIe*)/Compute Express Link (CXL)* devices and integrated accelerators, thus broadly encompassing the set of agents which read from and write to either caches or memory, excluding IA cores.

The Intel RDT feature set provides a series of monitoring and allocation capabilities such as Cache Monitoring Technology (CMT), Memory Bandwidth Monitoring (MBM), Cache Allocation Technology (CAT), Code and Data Prioritization (CDP), Memory Bandwidth Allocation (MBA) and others. These technologies enable monitoring and control of shared platform resources, such as the L3 cache capacity or main memory bandwidth, which may be in use by many applications, containers or VMs running on the platform concurrently. As described in subsequent chapters, these features enable deterministic behavior and fairness in communications, real-time and other usages, and are initially introduced in Section 1.3.

The Intel RDT features are based on a set of architectural tags, described in the following section, and fundamental capabilities for enabling monitoring and control over shared platform resources under the control of an operating system (OS) or virtual machine monitor (VMM), as described in the chapter on Reference Software Architecture.

1.1 High Level Usage Models

A wide variety of industry deployment models find value in either enhanced visibility into system resource utilization, or control over shared resources. As a result, a broad set of customer usage models are observed with Intel RDT, including but not limited to:

- **Cloud Hosting in the datacenter** – Prioritizing important Virtual Machines (VMs) and containing or mitigating “noisy neighbors”.
- **Public/Private Cloud** – Isolating an important infrastructure VM which provides networking services such as a VPN to bridge the private cloud to the public cloud.

- **Datacenter Infrastructure** – Protecting virtual switches which provide local networking.
- **Communications** – Ensuring consistent performance and containing background tasks on a network appliance built atop an Intel® Xeon® Server Platform.
- **Content Delivery Networks (CDNs)** – Prioritizing key parts of the content serving application in order to improve throughput.
- **Networking** – Containing the impact of consolidated or co-located containers to help reduce jitter and reduce packet loss in noisy scenarios, and protecting high-performance applications based on the Dataplane Development Kit (DPDK).
- **Industrial Control** – Prioritizing important sections of code to help meet real-time requirements.

Varying usage models drive differing requirements. Datacenter usages may require control over relative container prioritization and management of tail latencies, for instance, while industrial control usages may require strict management of control loop cycle times, including the use of model-specific extended Intel RDT features. A number of example use cases are described in more detail based on abstracted examples of real-world deployments in the chapter on Reference Software Architecture.

1.2 Scope

Broadly, this document discusses the following topics:

- An introduction to key Intel RDT architectural concepts and design philosophy.
- Details of architectural Intel RDT monitoring and allocation features for CPU agents and non-CPU agents.
- Details of model-specific Intel RDT monitoring and allocation features for CPU agents and non-CPU agents.
- Considerations for BIOS writers, and those consuming ACPI enumeration tables generated by BIOS.
- An overview of various real-world software usages of Intel RDT features that have been observed, and recommended software enabling strategies.

The following topics are not covered (or are covered in a limited context):

- Intel RDT for CPU Agents and non-CPU Agents architectural details - feature enumeration and interfaces using CPUID and configuration using MSRs. These details are provided in the Intel® 64 Architecture Software Developer's Manual (SDM), Volume 3B, Chapter Title: Debug, Branch Profile, TSC, and Intel® Resource Director Technology (Intel® RDT) Features [1], and the document entitled Intel® Architecture Instruction Set Extensions and Future Features [2], as applicable.

1.3 Audience

The intended audience for this specification includes Intel RDT consumers, users and implementers, across OS/VMM software, resource management driver and control loop developers, administrators, managers of datacenter infrastructure, workload owners and embedded and communications developers. Additionally, this specification may be of interest to those developing utilities, BIOS routines, administrative libraries and orchestration frameworks.

1.4 References

Table 1-1. References

Description
[1] Intel® 64 and IA-32 Architectures Software Developer's Manual. Volume 3B, Chapters 18.18 and 18.19. https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html
[2] Intel® Architecture Instruction Set Extensions and Future Features. Instruction Set Architecture (intel.com)
[3] Intel® Virtualization Technology for Directed I/O Specification. http://www.intel.com/content/www/us/en/embedded/technology/virtualization/vt-directed-io-spec.html
[4] Unified Extensible Firmware Interface Forum – Links to ACPI-Related Documents (incudes IRDT table title and signature). https://uefi.org/acpi
[5] PCIe Express Specification, v5.0 or newer. https://pcisig.com/specifications
[6] Compute Express Link Specification, v1.0 or newer. https://www.computeexpresslink.org/download-the-specification
[7] User space software for Intel® Resource Director Technology https://github.com/intel/intel-cmt-cat
[8] ACPI Software Programming Model https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/05_ACPI_Software_Programming_Model/ACPI_Software_Programming_Model.html#system-resource-affinity-table-srat
[9] Intel® Platform Monitoring Technology (Intel® PMT) - External Specification https://www.intel.com/content/www/us/en/content-details/710389

§

2 Intel® Resource Director Technology Overview

This chapter provides an overview of Intel® RDT features, including goals, key ingredients, and the architectural framework, which are discussed in more detail in the chapters that follow.

2.1 Common Tags

Intel RDT provides a layer of abstraction between applications and logical processors through the use of numeric tags. Both CPU agents and non-CPU agents use the following tags for resource monitoring and allocation, respectively:

- Resource Monitoring IDs (RMIDs) are used for monitoring of shared platform resource utilization.
- Classes of Service (CLOS) are used for control of shared platform resources, such as L3 cache occupancy or memory bandwidth.

The RMID and CLOS tags are described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. RMID and CLOS tags are independent. Usage of RMID tags does not affect CLOS, and vice versa (however, when CLOS tags are used to affect resource allocations, the effects may be observed with RMID-based monitoring features.) An RMID-based monitoring feature does not incur hardware overhead or affect a CLOS-based allocation feature. A product may be built to implement RMID-based monitoring features, CLOS-based control features, or both.

For CPU agents, RMIDs and CLOS tags are associated with the operation of a logical processor through the IA32_PQR_ASSOC MSR.

For non-CPU agents, a series of MMIO interfaces is used to associate upstream traffic from I/O devices with RMID and CLOS tags, and the numerical interpretation of the tags is the same as for processor traffic. (For example, the RMID value "5" used to track processor thread resource consumption means the same thing as when the RMID value "5" is used to track the cache fill behavior of a PCIe device.) These MMIO interfaces for tagging non-CPU agents are discovered using an ACPI structure called I/O Intel RDT, that is, IRDT. (see Chapter 5.)

Other features may utilize RDT tags, such as Resource Monitoring IDs, to track and report other telemetry events in the processor. Examples include per-RMID telemetry available both in-band and out-of-band as specified in the Intel® Platform Monitoring Technology (Intel® PMT) specification [9] and associated platform-specific telemetry events lists.

2.2 Enumeration of Supported Features

Software enumeration of supported RDT features is enabled through the CPUID instruction for CPU-centric features, and through the Advanced Configuration and Power Interface (ACPI) for platform-centric features.

Further enumeration details of CPUID-enumerated features, including Hybrid processor support, are provided in the Intel® 64 Architecture Software Developer's Manual (SDM), Volume 3B, Chapter Title: *Debug, Branch Profile, TSC, and Intel® Resource Director Technology (Intel® RDT) Features* [1] and the document entitled Intel® Architecture Instruction Set Extensions and Future Features [2], as applicable.

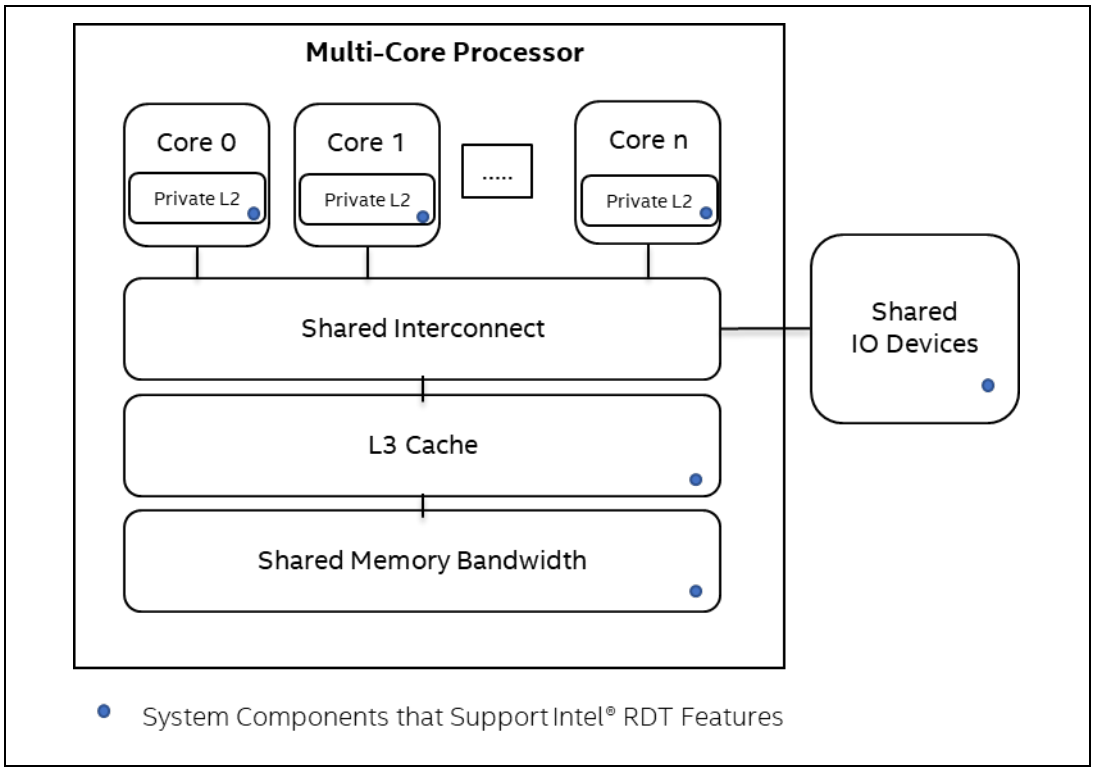
Details on ACPI-enumerated features are provided in subsequent sections of this document.

2.3 L3 Configurations

This specification describes two types of high level L3 configurations that may support Intel RDT features:

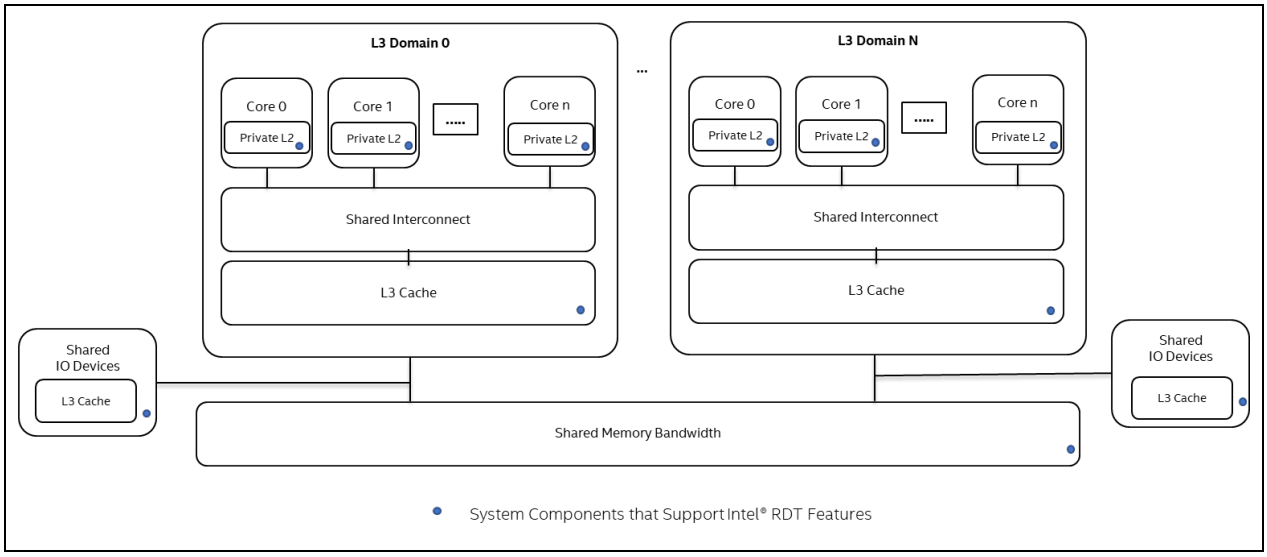
1. **Shared-L3 Configuration:** There is a common shared L3 cache for all the agents in the SoC, as shown in Figure 2-1. This SoC configuration supports interfaces for Intel RDT features based on the CPUID instruction for feature enumeration and Model-Specific Registers (MSRs) for feature configuration and telemetry retrieval.

Figure 2-1. Shared-L3 Configuration System Model and Presence of Intel® RDT Features



2. **Multiple-L3 Configuration:** There may be more than one L3 cache instances that are local to CPU Agents or non-CPU Agents respectively, as shown in Figure 2-2.

Figure 2-2. Multiple-L3 Configuration System Model and Presence of Intel ® RDT Features



A set of features defined within a particular cache domain, such as an L3 cache supporting a number of logical processors, may be referred to as a Resource

Telemetry Domain (RTD, for monitoring features) or a Resource Allocation Domain (RAD, for allocation features). More generally, a resource which supports Intel RDT monitoring features, allocation features or both may be referred to as a Resource Management Domain (RMD).Figure 2-2 shows an example of multiple RMDs.

See Appendix A.4 for Intel RDT feature mapping for CPU agents and non-CPU agents in different SoC configurations.

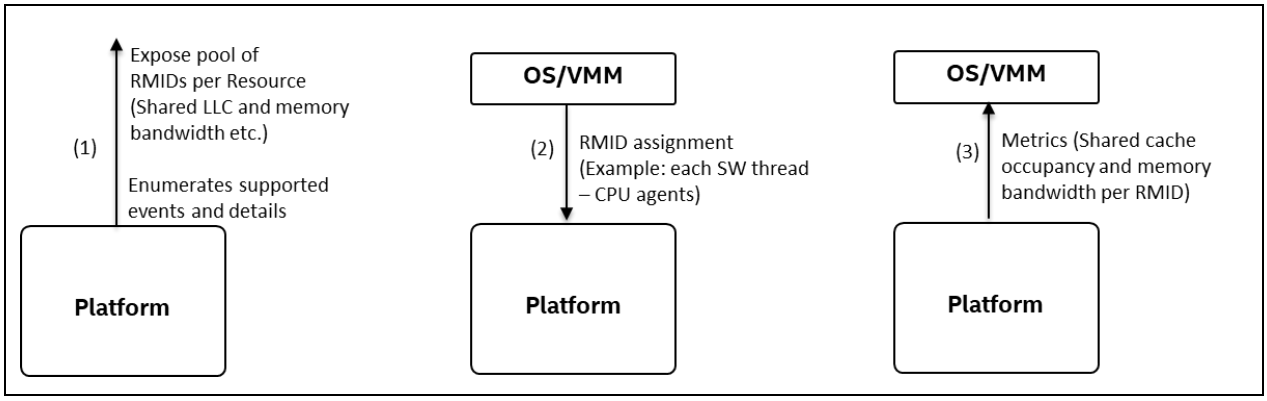
2.4 Intel® RDT Monitoring Technologies

2.4.1 Intel® RDT Monitoring Key Ingredients

Intel RDT Monitoring enables monitoring shared platform resources, such as L3 cache occupancy and memory bandwidth, based on software-defined Resource Monitoring IDs (RMIDs) that are tagged to applications or VMs on a per-thread basis (Figure 2-3). For CPU Agents, each logical processor exposes the IA32_PQR_ASSOC MSR to allow the OS/VMM to specify an RMID when an application, thread or VM is scheduled on a core.

Resource monitoring for the indicated application/thread/VM is then performed by hardware based on the RMID with which it is associated, and software can read back the L3 cache occupancy for a given RMID via counter registers (if the CMT feature is supported for instance). Each thread of an application may be tracked with a distinct RMID, or threads may be grouped into a single RMID, based on the granularity of monitoring required. Threads within a VM, apps within a VM, entire VMs or groups of VMs can similarly be tracked with RMIDs with variable granularity as needed.

Figure 2-3. Intel® RDT Monitoring – Enabling RMID-Based Monitoring for Shared Resources



The basic ingredients of Intel RDT Monitoring are as follows:

- CPUID and/or ACPI constructs to indicate support for Intel RDT Monitoring and sub-features (CMT, MBM, and so on) for Resource Telemetry Domains (RTD).

- Enumeration of the total number of RMIDs that can be tracked in the given RTD.
- Mechanisms to allow system software (OS/VMM) to specify the RMID of software threads and non-CPU agents.
- Mechanisms to allow system software to retrieve collected metrics on a per-RMID basis via architectural MSRs or MMIO interfaces.

The first ingredient to make use of Intel RDT Monitoring is to enumerate the set of monitoring capabilities provided on the given Resource Management domain via CPUID or ACPI and determine the number of RMIDs available for tracking on a particular Resource Telemetry Domain (RTD, that is, caching domain). This will allow the OS/VMM to determine how many unique IDs it may use. Given that certain processor topologies may include heterogenous capabilities which vary per-processor, it is recommended that software enumerate Intel RDT CPUID leaves from the perspective of each logical processor (LP) to construct the list of supported capabilities and which resources (such as L3 cache) may be shared among various LPs.

The second ingredient (Intel RDT Monitoring association) allows the OS/VMM to specify the RMID of the running software thread to the platform for CPU agents. The OS/VMM can also specify the RMID for upstream traffic and operation of non-CPU agents.

The third ingredient (Intel RDT marking and associated hardware support) enables each memory request from the CPU agents and non-CPU agents to be tagged with the RMID provided by the OS/VMM.

The fourth ingredient is Intel RDT Monitoring reporting. When the monitoring data retrieval register is programmed with the RMID and the specific event code of interest (L3 Cache Occupancy for example), this information is appropriately retrieved and provided back.

Multiple Intel RDT Monitoring features may exist within a platform, but the software should not assume that the presence of one Intel RDT Monitoring feature implies the existence of any others. Intel RDT features are independently enumerated in the sequence described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, Section 18.18.4, in order to avoid ambiguous situations.

2.4.2 Shared-L3 versus Multiple-L3 Configuration

Intel RDT Monitoring features may have different scope definitions depending on L3 configuration. With the shared-L3 configuration, CPU agents and non-CPU agents allocate into a shared L3 cache. Hence, all monitoring features have a consistent definition for CPU agents and non-CPU agents.

With the multiple-L3 configuration, non-CPU agents may have a separate nearby L3 cache which is distinct from CPU agents' L3 cache. Hence, monitoring features may have different definitions for CPU agents and non-CPU agents. For example, in certain implementations, non-CPU agents with a near

L3 cache implementation may report memory bandwidth monitoring data from the near cache only.

2.5 Intel® RDT Allocation Technologies

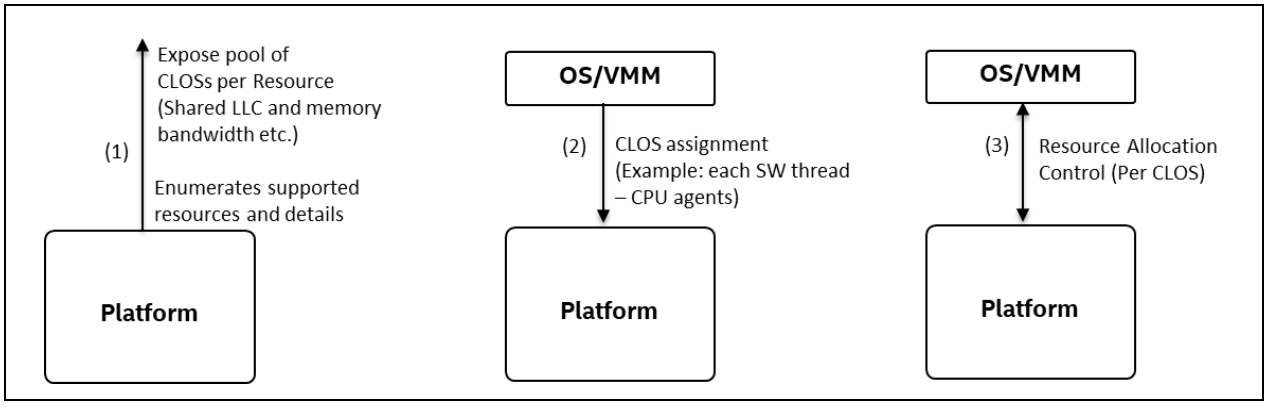
2.5.1 Intel® RDT Allocation Key Ingredients

Intel RDT Allocation enables resource allocation based on Class of Service (CLOS) tags. The processor exposes Classes of Services into which applications (or individual threads) and traffic from I/O devices may be assigned. A CLOS may have multiple associated resource allocation properties. For example, there may exist controls for each CLOS to specify L2 capacity available to that CLOS, L3 capacity available, memory bandwidth available, and other properties (Figure 2-4).

In the case of L3 capacity control features, for instance, such as Cache Allocation Technology (CAT), the cache allocation for a given thread is restricted based on the class with which they are associated. Similarly, in certain implementations supporting non-CPU agent controls, context-associated and upstream traffic from I/O devices may be controlled as it utilizes shared system resources. Each CLOS can be configured using bitmasks which represent capacity, and the degree of overlap and isolation between classes in allocation features which influence the SOC caches.

For CPU agents, each logical processor exposes the IA32_PQR_ASSOC MSR to allow the OS/VMM to specify a CLOS when an application, thread or VM is scheduled. Cache Allocation for the application/thread/VM is then controlled based on the CLOS and the associated bitmask.

Figure 2-4. Intel® RDT Allocation – Enabling CLOS-based Allocation for Shared Resources



The basic ingredients of Intel RDT Allocation are as follows:

- CPUID or ACPI constructs to indicate whether Intel RDT Allocation and sub-features (CAT, MBA, and so on) for Resource Allocation Domains (RADs) are supported and enumerate the total number of CLOS that may be associated to shared platform resources on the platform.

- Mechanisms to allow system software (OS/VMM) to specify the CLOS of software threads and non-CPU agents.
- Mechanisms to allow system software to configure the shared platform resource levels available to each CLOS via architectural MSRs or MMIO interfaces.

The first ingredient to make use of Intel RDT Allocation is to enumerate the level of allocation capability provided on the given Resource Allocation Domain via CPUID and/or ACPI and determine the number of CLOSs available for allocating shared platform resources on a particular RAD (that is, a certain L3 caching domain). This will allow the OS/VMM to determine how many unique IDs it may use. Given that certain processor topologies may include heterogenous capabilities which vary per-processor, it is recommended that software enumerate Intel RDT CPUID leaves from the perspective of each logical processor (LP) to construct the list of supported capabilities and which resources (such as L3 cache) may be shared among various LPs.

The second ingredient (Intel RDT Allocation association) allows the OS/VMM to specify the CLOS of the running software thread to the platform for CPU agents. The OS/VMM can also specify the CLOS for upstream traffic and operation of non-CPU agents.

The third ingredient (Intel RDT marking and associated hardware support) enables each memory request from CPU agents and non-CPU agents to be tagged with the CLOS provided by the OS/VMM.

The fourth ingredient is Intel RDT Allocation control, when the allocation register is programmed with the CLOS and allocation control is performed by the specific shared platform resource (L3 Cache capacity for example).

Multiple Intel RDT Allocation features may exist within a platform. The software should not assume that the presence of one RDT Allocation feature implies the existence of any others. Intel RDT features are independently enumerated in the sequence described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, in order to avoid ambiguous situations.

2.5.2 Shared-L3 versus Multiple-L3 Configuration

Intel RDT Allocation features may have different definitions depending on L3 configuration. With the shared-L3 configuration, CPU agents and non-CPU agents allocate into a shared-L3 cache. Hence, all allocation features have a consistent definition for CPU agents and non-CPU agents. With the multiple-L3 configuration, non-CPU agents may have a separate near L3 cache which is different from the CPU agents' L3 cache. Hence, allocation features may have different definitions for CPU agents and non-CPU agents. For example, non-CPU agents with a near L3 cache implementation provide separate interfaces for cache capacity allocation for the near L3 cache.

Chapter 3 and Chapter 4 provide details about each Intel RDT Monitoring and Allocation features for CPU agents and non-CPU agents.

§

3 Intel® Resource Director Technology for CPU Agents

This chapter contains an overview of the Intel RDT features for CPU agents. Chapter 4 describes details about features for non-CPU agents.

3.1 Intel® RDT Monitoring Features

The Intel RDT Monitoring architecture enables monitoring of the utilization level of critical shared platform resources and provides this data directly to the Hypervisor, Operating System or other privileged software. Intel RDT Monitoring supports three event codes: 1) L3 cache occupancy 2) L3 Total External bandwidth 3) L3 Local External bandwidth. This allows more efficient scheduling based on resource use, as well as application tuning and performance prediction based on resource use characterization, and optionally better reporting and billback. This functionality complements Intel RDT Allocation, which provides control over shared platform resources available to CPU agents.

3.1.1 Common Framework

The following mechanisms are shared by Intel RDT Monitoring features:

- CPUID feature bits to enumerate the presence of the Intel RDT Monitoring capabilities and the details of each sub feature.
- The IA32_PQR_ASSOC MSR, which the OS or Hypervisor uses to specify the RMID for each software thread scheduled to run on a logical processor. See Figure 3-2.
- The IA32_QM_EVTSEL and IA32_QM_CTR MSRs, to read cache occupancy and bandwidth statistics. See Figure 3-3.

Software may flexibly associate RMIDs with threads, applications, VMs, or containers. (See Figure 3-1). If multiple logical processors within a Resource Telemetry Domain (RTD) are assigned the same RMID, the total resource monitoring telemetry by these logical processors will be accumulated together and the total reported by hardware.

Monitoring data is retrieved using a window-based interface. Software writes an event ID and RMID to the IA32_QM_EVTSEL MSR and hardware provides the resulting data back in the IA32_QM_CTR MSR.

Refer to Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, for details on CPUID and MSR usage.

Figure 3-1. Resource Monitoring IDs (RMIDs) Assignment Flow

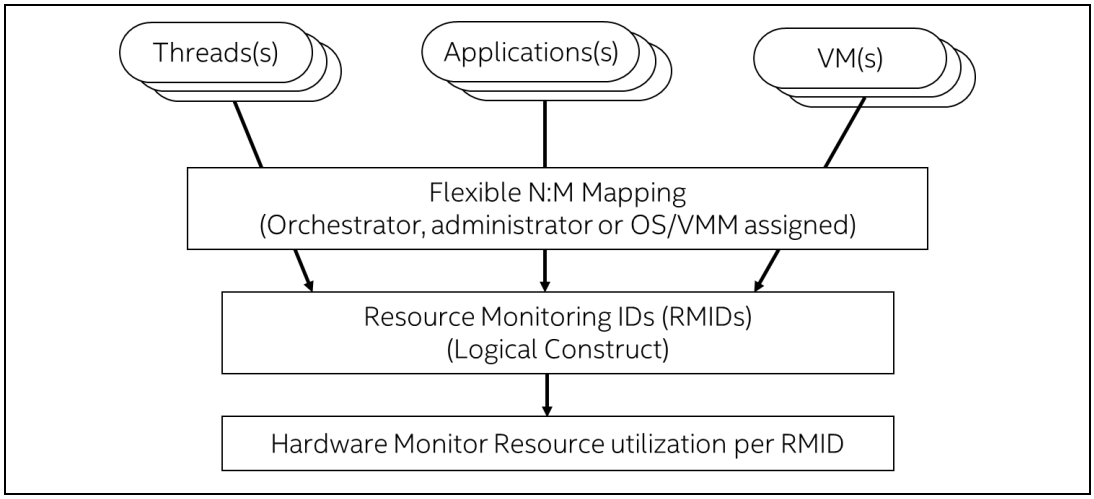


Figure 3-2. IA32_PQR_ASSOC MSR to Set RMID

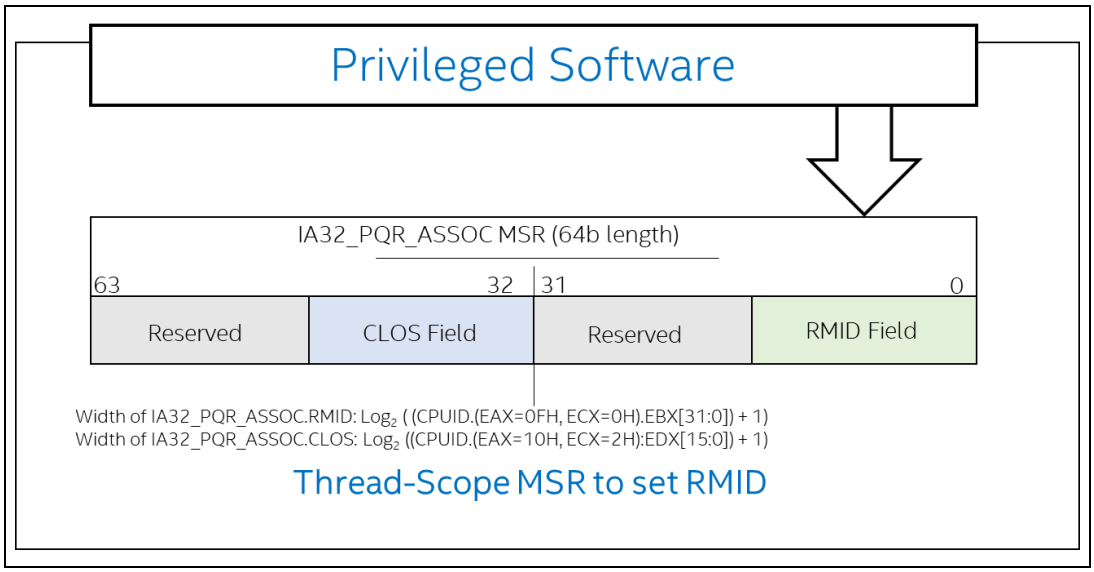
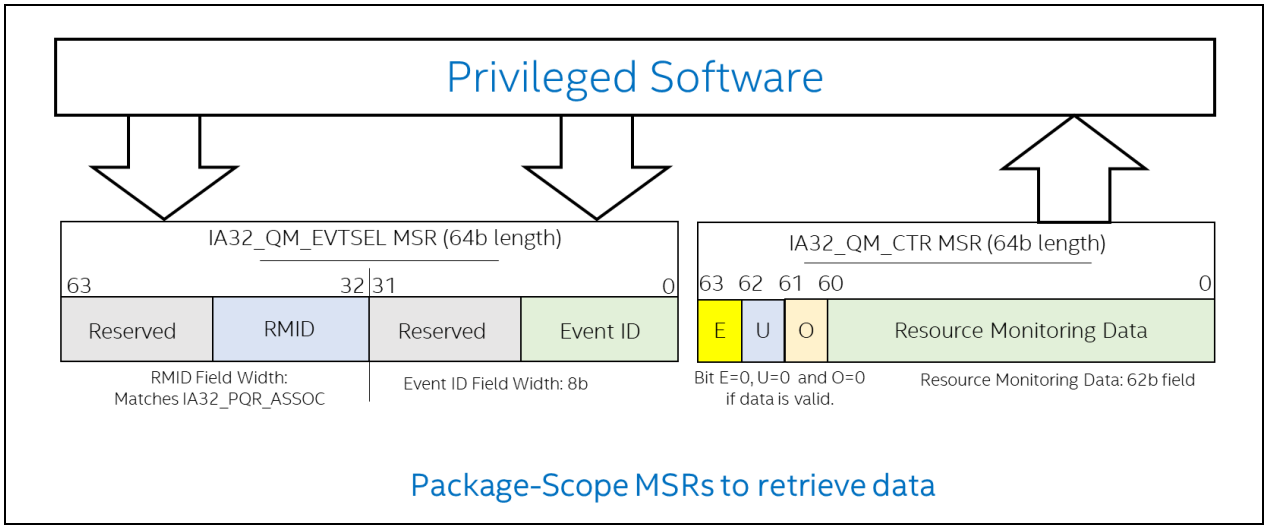


Figure 3-3. IA32_QM_EVTSEL and IA32_QM_CTR MSRs



3.1.2 Memory Regions

Certain processors support a system-level enumeration of Memory Regions, which are part of the common infrastructure used in the RDT feature set. Other feature sets may also make use of the Memory Region definition which RDT establishes.

A Memory Region is defined as comprising one or more physically-addressed memory ranges. Certain processors support a system-level enumeration of Memory Regions.

Multiple Memory Regions may be defined by the platform to independently describe physical addresses backed by a particular type of memory, which may exhibit varying capacity, latency, bandwidth, and locality characteristics. Examples includes DRAM or CXL-attached memories, whether attached locally or to a different processor over a coherent interconnect link.

The Memory Regions populated on a particular processor are described by the system BIOS in the ACPI Memory Range and Region Mapping (MRRM) table, which is described in Chapter 5, *BIOS Considerations*. This information may be combined with other ACPI tables, such as HMAT, SRAT and CEDT, to gain more insight regarding memory types connected to a certain processor.

For a modern platform, it becomes advantageous for the processor to provide a capability to directly measure and allocate memory bandwidth across these multiple memory regions simultaneously. Such processor capabilities, when enabled, allow software to gather usage telemetry, adjust Memory Bandwidth Allocation (MBA) policies, and build control loops to ensure performance goals are met. As described below, Intel provides these capabilities as Region-Aware Memory Bandwidth Monitoring and Allocation which are described in their respective sections.

3.1.3 Cache Occupancy Monitoring Technology

Intel RDT Cache Occupancy Monitoring Technologies provide visibility into cache utilization. Features such as Cache Monitoring Technology (CMT) provide occupancy counters on a per-RMID basis such that cache occupancy by each RMID may be tracked and read back in real-time during system operation.

More specific feature details about CMT are provided in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. Note that the MSRs are listed in Appendix A.5. See Appendix A.2 for CMT feature supported product details.

3.1.3.1 L3 Cache Monitoring Technology

L3 Cache Monitoring Technology (CMT) allows an Operating System, Hypervisor or similar system management agent to determine the usage of L3 cache of the Resource Telemetry Domain (RTD) by applications running on the platform.

3.1.4 Memory Bandwidth Monitoring

Memory Bandwidth Monitoring (MBM) provides monitoring of bandwidth from one level of the cache or resource hierarchy to the next, allowing bandwidth-aware scheduling decisions, inter-RTD scheduling optimization, and enabling feedback to bandwidth allocation features which allow control over memory bandwidth.

More specific feature details about MBM are provided in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. Note that the MSRs are listed in Appendix A.5. See Appendix A.2 for MBM feature supported product details.

3.1.4.1 L3 Total and Local External Memory Bandwidth Monitoring

L3 Total and Local External Memory Bandwidth Monitoring allows system software to monitor the use of bandwidth between L3 cache and local or remote memory. In certain implementations, MBM is not guaranteed to track directory, Extended Prediction Table (XPT) prefetcher or related types of traffic.

3.1.4.2 Region Aware Memory Bandwidth Monitoring

The Region Aware Memory Bandwidth Monitoring (MBM) feature provides a set of counters simultaneously indexed by RMID and Region to measure the memory bandwidth utilization of an RDT Resource Monitoring ID (RMID – typically mapped to software threads, applications, containers or virtual machines) to a Memory Region in the system. Typical hardware feature support for Region Aware MBM includes the ability to independently track many RMIDs simultaneously accessing several Memory Regions. Unlike prior generations of MBM, the Region Aware MBM feature primarily uses an MMIO-based interface.

Software may consult the Enhanced Resource Director Technology (ERDT) ACPI table for enumeration of specific capabilities of this feature on a given processor generation. The ERDT table defined in Chapter 5, *BIOS Considerations,* provides information regarding capabilities and architectural parameters such as the number of RMIDs supported. See Chapter 6, *MMIO Register Descriptions* for details of the register interfaces used.

3.2 Intel® RDT Allocation Features

The Intel RDT Allocation architecture enables control over utilization level of critically shared platform resources and provides this control directly to the Hypervisor or Operating System. This allows more efficient resource usage as well as application prioritization and determinism restoration based on resource repartitioning. The implementation of Intel RDT Allocation features may be product-specific or architectural. These capabilities complement Intel RDT monitoring, which provides insight into shared platform resource utilization by CPU agents.

3.2.1 Common Framework

The following mechanisms are shared by Intel RDT allocation features:

- CPUID feature bits to enumerate the presence of Intel RDT Allocation capabilities and the details of each sub feature.
- The IA32_PQR_ASSOC MSR which software uses to specify the CLOS for each software thread. See Figure 3-5.
- Mechanisms in hardware to specify resource usage to apply to each Class of Service.

Software can flexibly associate Classes of Service with threads, applications, VMs, or containers (see Figure 3-4). CLOS values are shared across all allocation features. A particular numeric CLOS value has the same meaning from the viewpoint of all cores. Each CLOS has an associated set of mask registers as described later to associate that CLOS with a fraction of the shared platform resources. If multiple logical processors within a Resource Allocation Domain (RAD) are assigned the same CLOS, then resource allocations associated with that CLOS will be shared among that set of logical processors.

Figure 3-4. Classes of Service (CLOS) Association Flow

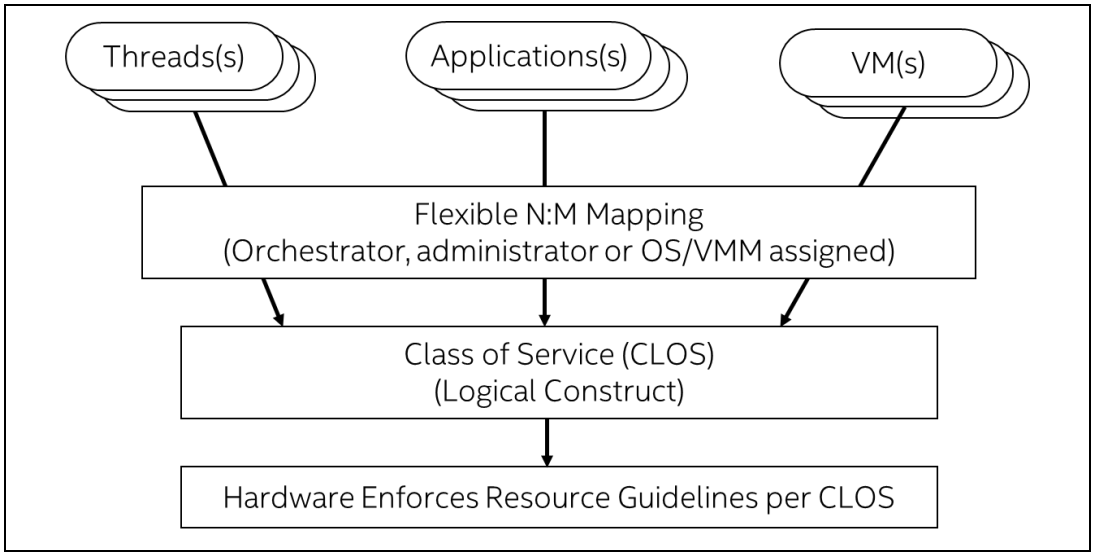
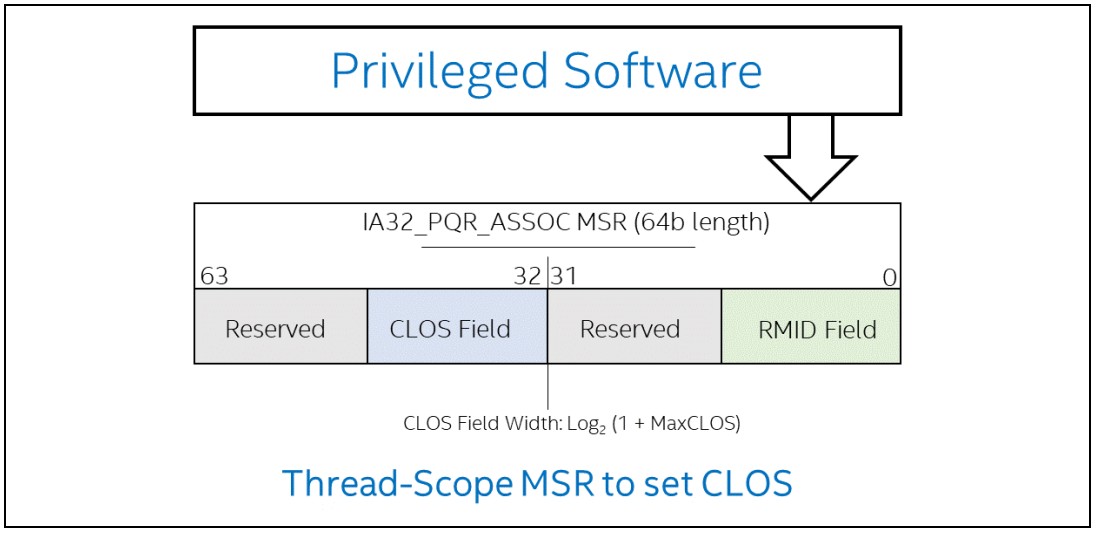


Figure 3-5. The IA32_PQR_ASSOC MSR to Set CLOS



For each resource, a block of registers is defined for software to configure the allocation values for each CLOS. The definition of the register fields depends on the type of resource being managed and is discussed in subsequent sections.

3.2.2 Memory Regions

See Section 3.1.2 for a discussion of Memory Regions which are a shared infrastructure component used across both RDT Allocation and RDT Monitoring technologies, in particular Region Aware MBM and Region Aware MBA.

3.2.3 Cache Occupancy Allocation Technologies

A family of Cache Occupancy Allocation Technologies allows control over shared cache space on a per-CLOS basis, enabling configurable isolation or overlap for potentially improved throughput, fairness, determinism and/or differentiation. These features are known as Cache Allocation Technology (CAT), which is the term used in this document. Certain processors may support architectural or model-specific forms of CAT depending on the product generation. Model-specific implementations are discussed in Appendix B.1.4.

More specific feature details about CAT are provided in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. Note that the MSRs are listed in Appendix A.5. See Appendix A.2 for CAT feature supported product details.

3.2.3.1 L2 Cache Allocation Technology

L2 Cache Allocation Technology (L2 CAT) allows system software to specify the amount of L2 cache space of the Resource Allocation Domain into which an application can fill.

3.2.3.2 L2 Cache Code and Data Prioritization

L2 Code Data Prioritization (L2 CDP) provides differentiation between code and data for L2 cache usage by a single Class of Service. In a case where an application has a large code footprint which can overwhelm data in the cache, or vice versa, the ability to separately prioritize code and data is valuable.

L2 CDP provides a pair of allocation bitmasks for each Class of Service (rather than a single bitmask per CLOS as in L2 CAT), to allow system software to independently configure the amount of L2 cache available to code and data.

3.2.3.3 L3 Cache Allocation Technology

L3 Cache Allocation Technology (L3 CAT) allows an Operating System (OS), a Hypervisor, Virtual Machine Manager (VMM), or similar system service management agent to specify the amount of L3 cache space within a Resource Allocation Domain (RAD) into which a CLOS may fill.

3.2.3.4 L3 Cache and Data Prioritization

L3 Code Data Prioritization (L3 CDP) provides differentiation between code and data for L3 usage by a single Class of Service. In a case where an application has a large code footprint which can overwhelm data in the cache, or vice versa, the ability to separately prioritize code and data is valuable.

L3 CDP provides a pair of allocation bitmasks for each Class of Service (rather than a single bitmask per CLOS as in L3 CAT), to allow system software to independently configure the amount of L3 cache available to code and data.

3.2.4 Memory Bandwidth Allocation

Memory Bandwidth Allocation (MBA) allows the system software to control access bandwidth to memory. It allows slowing "noisy neighbor" threads which may be overutilizing bandwidth and enables the creation of closed-loop control systems (monitoring and control combined) by exposing control over a credit-based throttling mechanism.

More specific feature details about MBA are provided in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. Note that the MSRs are listed in Appendix A.5. See Appendix A.2 for MBA feature supported product details.

There are multiple generations of MBA, each extending additional capabilities:

1. **First Generation MBA (Interface Scope)** – This is the initial implementation of the MBA feature which provides indirect and approximate control over memory bandwidth available per-core. See Section 3.2.3.1 for implementation details and see the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, for legacy enumeration, interface and per-CLOS delay value resolution details.
2. **Second Generation MBA (Interface Scope)** - This enhanced MBA capability provides improved efficiency and accuracy in bandwidth control, along with providing increased system throughput. Rather than a strict bandwidth control mechanism, a dynamic hardware controller is implemented, which can react to changing bandwidth conditions at the microsecond level. Before using the second-generation MBA feature, the MBA hardware controller requires a BIOS-assisted calibration process that may include inputs such as the number of memory channels populated and other system parameters; this is a change from the first generation of MBA.
   Intel's BIOS reference code includes a default configuration that is recommended for general usage, and BIOS profiles may be created with alternate tuning values to optimize for certain usages (such as stricter bandwidth control). See Section 3.2.3.2 for implementation details and Intel® 64 and the IA-32 Architectures Software Developer's Manual, Volume 3B, for legacy enumeration and interface details.
3. **Third Generation MBA (Agent Scope)** - The third generation MBA feature on future processors based on the codename Granite Rapids microarchitecture further enhances MBA with per-logical-processor control and a further improved controller design. Total memory bandwidth (all L3 miss traffic) is now managed by MBA. This implementation follows the prior MBA precedent of delivering significant enhancements without a major software overhaul, and while preserving backward compatibility. See Section 3.2.3.3 for implementation details and the Intel® 64 and the IA-32 Architectures Software Developer's Manual, Volume 3B, for legacy enumeration and interface details.

MBA performance properties change over time, for instance enhancing system-level efficiency. Software should not assume that performance properties or specific tunings of MBA remain identical across product generations. Third generation MBA shifts from interface-scope to agent-scope bandwidth control

support, and scheduler re-tuning to take advantage of this enhancement may be beneficial. Legacy architectural implementations of MBA are enumerated in the sequence described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, in order to avoid ambiguous situations.

The MBA feature provides the following architectural components:

- A mechanism to enumerate the MBA capability to control the bandwidth from each level of the cache (for example, L2, L3) to the next level.
- A mechanism for the OS or Hypervisor to configure the amount of bandwidth available to a particular Class of Service via a bandwidth control (throttling) value as discussed later.
- Mechanisms for the OS or Hypervisor to specify the Class of Service to which a thread belongs.
- Hardware mechanisms to guide and enforce the delay value at each level of the cache hierarchy when an application has been designated to belong to a specific Class of Service.

Note that in some usages such as those seeking bandwidth control in MB/s, MBA may require either application-level performance feedback or complementary Memory Bandwidth Monitoring (MBM) to use in the most optimal way. Backward compatibility of the software interfaces is preserved, and enhanced MBA generational changes manifest as enhancements atop the MBA feature baseline.

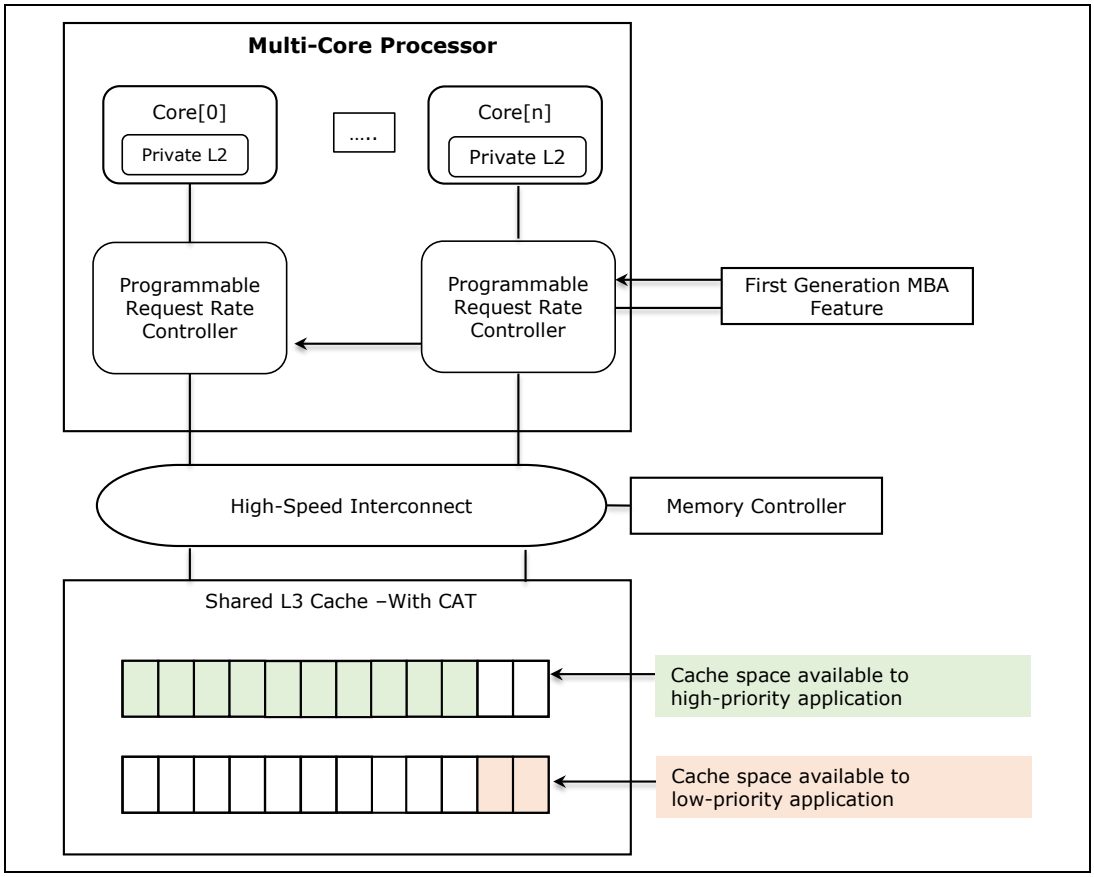
3.2.4.1 First Generation Memory Bandwidth Allocation

The Memory Bandwidth Allocation (MBA) feature provides indirect and approximate control over memory bandwidth available per-core and was introduced on the Intel® Xeon® Scalable Processor Family. This feature provides a method to control applications which may be over-utilizing bandwidth relative to their priority in environments such as the datacenter.

The MBA feature uses existing constructs from the Intel RDT feature set including Classes of Service (CLOS). A given CLOS used for L3 CAT for instance means the same thing as a CLOS used for MBA. Infrastructure such as the MSR used to associate a thread with a CLOS (the IA32_PQR_ASSOC_MSR) and some elements of the CPUID enumeration (such as CPUID leaf 10H [Cache Allocation Technology Enumeration Leaf]) are shared.

The high-level implementation of Memory Bandwidth Allocation is shown in Figure 3-6.

Figure 3-6. A High-Level Overview of the First-Generation MBA Feature



As shown here, the MBA feature introduces a programmable request rate controller between the cores and the high-speed interconnect, enabling indirect control over memory bandwidth for cores over-utilizing bandwidth relative to their priority. For instance, high-priority cores may be run un-throttled, but lower priority cores generating an excessive amount of traffic may be throttled to enable more bandwidth availability for the high-priority cores.

Because the MBA uses a programmable rate controller between the cores and the interconnect, higher-level shared caches and memory controller, bandwidth to these caches may also be reduced, so care should be taken to throttle only bandwidth-intense applications which do not use the off-core caches effectively.

The bandwidth control (throttling) values exposed by MBA are approximate and are calibrated to specific traffic patterns. As workload characteristics vary, the bandwidth control values provided may affect each workload differently. In cases where precise control is needed, the Memory Bandwidth Monitoring (MBM) feature can be used as input to a software controller which makes decisions about the MBA bandwidth control level to apply.

Legacy enumeration and configuration details are discussed in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.

3.2.4.1.1 Usage Considerations

As the memory bandwidth control that MBA provides is indirect and approximate, using the feature with a closed-loop controller to also monitor memory bandwidth and how effectively the applications use the cache (via the Cache Monitoring Technology feature) may provide additional value. This approach also allows administrators to provide a bandwidth target or set point which a controller could use to guide MBA bandwidth control (throttling) values applied, and this allows bandwidth control independent of the execution characteristics of the application.

As control is provided per processor core (the max of the delay values of the per-thread CLOS applied to the core), the user should take care in scheduling threads so as to not inadvertently place a high-priority thread (with zero intended MBA throttling) next to a low-priority thread (with MBA throttling intended), which would lead to inadvertent throttling of the high-priority thread, as the maximum resolved throttling value is applied.

3.2.4.2 Second Generation Memory Bandwidth Allocation

The second generation of Memory Bandwidth Allocation (MBA) is implemented in the 3rd Gen Intel® Xeon® Scalable Processor Family, and related Intel Atom® processors such as the P5000 Series. This enhanced MBA capability provides improved efficiency and accuracy in bandwidth control, along with providing increased system throughput. Rather than a strict bandwidth control mechanism, a dynamic hardware controller is implemented, which can react to changing bandwidth conditions at the microsecond level.

Before using the second-generation MBA feature, the MBA hardware controller requires a BIOS-assisted calibration process that may include inputs such as the number of memory channels populated and other system parameters; this is a change from the first generation of MBA. Intel BIOS reference code includes a default configuration that is recommended for general usage, and BIOS profiles may be created with alternate tuning values to optimize for certain usages (such as stricter bandwidth control) as described in the subsequent BIOS Considerations chapter.

Second generation MBA moves from static bandwidth control (throttling) at the core/uncore interface, to a more dynamic control method based on a hardware controller that tracks actual main memory bandwidth. This allows software that uses primarily the L3 cache to observe increased throughput for a given bandwidth control level, or fine-grained throughput benefits for software that exhibits L3-bound phases. Due to the closer consideration of memory bandwidth loading, this enhancement may lead to an increase in system efficiency when using second generation MBA relative to prior implementations of the feature. Backward compatibility of the software interfaces is preserved, and second-generation MBA changes manifest as enhancements atop the MBA feature baseline.

As with the prior generation feature, the second generation MBA uses CPUID for enumeration and throttling is performed using a mapping created from software thread-to-CLOS (in the IA32_PQR_ASSOC MSR), which is then

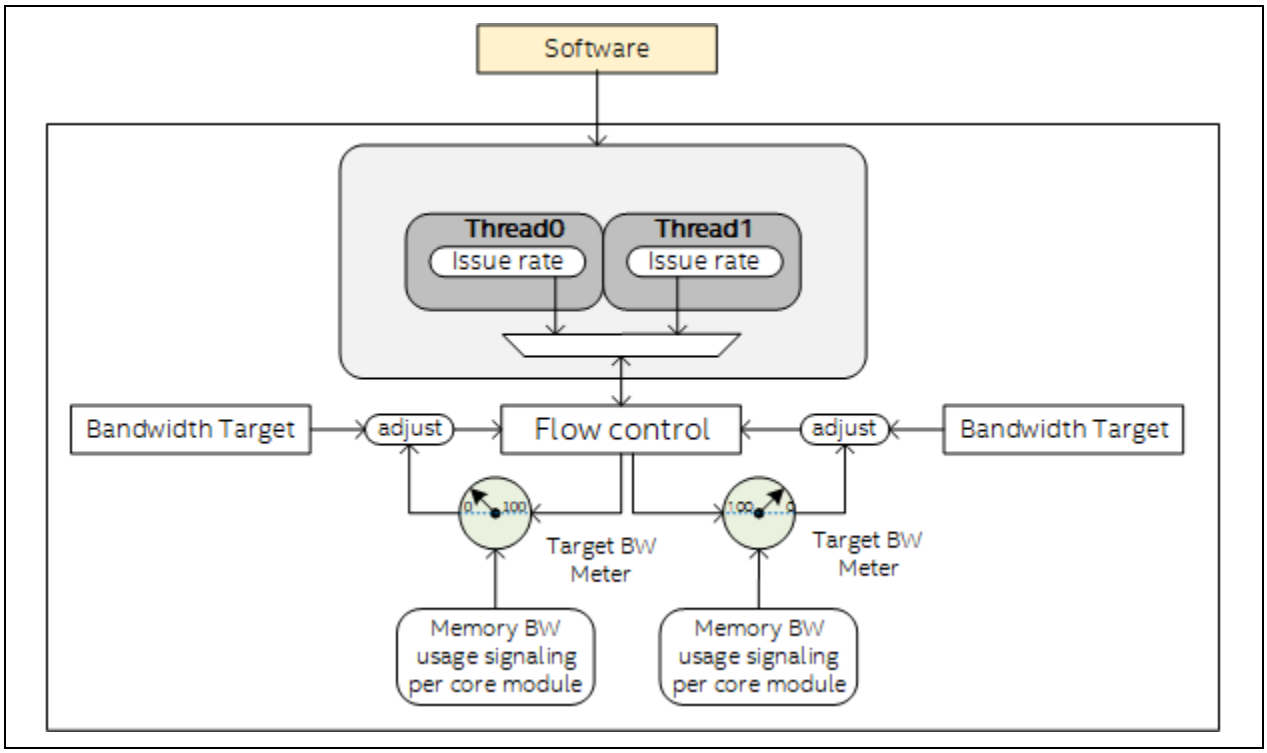
mapped per-CLOS to delay values via the IA32_L2_QoS_Ext_BW_Thrtl_n MSRs. A privileged operating system or virtual machine manager software may specify a per-CLOS delay value, 0-90% bandwidth throttling for instance, though the max and granularity values are platform dependent and enumerated in CPUID.

3.2.4.2.1 Second Generation MBA Advantages

Additional features added over first generation MBA are described next:

1. Previously, only the maximum delay value across two CLOS on a physical core could be selected in MBA. Second generation MBA allows a minimum delay value to be selected instead, which may enhance usage with Intel® Hyper-Threading Technology.
2. Only a single preprogrammed calibration table was possible in first generation MBA, meaning different memory configurations had the potential for different linearity and percent delay value error values depending on the configuration. This is addressed by the BIOS support in the second generation of MBA, and certain BIOS implementations may program a different calibration table per memory configuration, for instance.
3. The second-generation MBA controller provides the ability to more closely monitor the memory bandwidth loading and deliver more optimal results.
4. The new MBA hardware controller reduces the need for a fine-grained software controller to manage application phases for optimal efficiency. Note that a software controller may still be valuable to translate MBA bandwidth control values to bandwidths in GB/s or application Service Level Objectives (SLOs), such as performance targets.

Figure 3-7. Second Generation MBA, Including a Fast-Responding Hardware Controller



The second-generation MBA implementation is shown in Figure 3-7. The feature operates through the use of an advanced hardware controller and feedback mechanism, which allows automated hardware monitoring and control around the user-provided delay value set point. This set point and associated bandwidth control (throttling) value infrastructure remains unchanged from prior generation MBA, preserving software compatibility.

MBA enhancements, in addition to the new hardware controller, include:

1. Configurable delay selection across threads.
   - MBA 1.0 implementation statically picks the max MBA Throttling Level (MBAThrotLvl) across the threads running on a core (by calculating value = max(MBAThrotLvl(CLOS[thread0]), MBAThrotLvl(CLOS[thread1]))).
   - Software may have the option to pick either maximum or minimum delay to be resolved and applied across the threads; maximum value remains the default.
2. Increasing CLOS IDs from 8 to 15 in certain implementations (product-specific, see CPUID)
   - Previous certain implementations of the feature provided 8 CLOS tags for MBA.
   - The 3rd Gen Intel® Xeon® Scalable Processor Family and related Intel Atom® processors, such as the P5000 Series, increase this value to 15 (also consistent with L3 CAT).

3.2.4.2.2 Software-Visible Changes

A new model specific MSR is introduced with second generation MBA to allow software to select from the maximum (default) or minimum of resolved bandwidth control (throttling) values (see the previous formula). This capability is controlled via a bit in the new MBA_CFG MSR, shown in Table 3-1.

Table 3-1. MBA_CFG MSR Definition

Register Address		Architectural MSR Name / Bit Fields	Description
Hex	Decimal		
C84H	3204	MBA_CFG	MBA Configuration Register
		0	Min (1) or max (0) of per-thread MBA delays.
		63:1	Reserved. Attempts to write to reserved bits result in a #GP(0).

Note that bit[0] for min/max configuration is supported in second generation MBA but is removed (and reverts to reserved) in the third generation MBA feature when the controller logic becomes capable of managing bandwidth control values on a per-logical-processor or per-agent basis. The transient nature of this enhancement is why the min/max control remains model specific.

To enumerate and manage support for the model-specific min/max feature, software may use processor family/model/stepping as listed in Appendix A to match supported products, then CPUID to later detect enhanced third generation MBA support.

3.2.4.3 Third Generation Memory Bandwidth Allocation

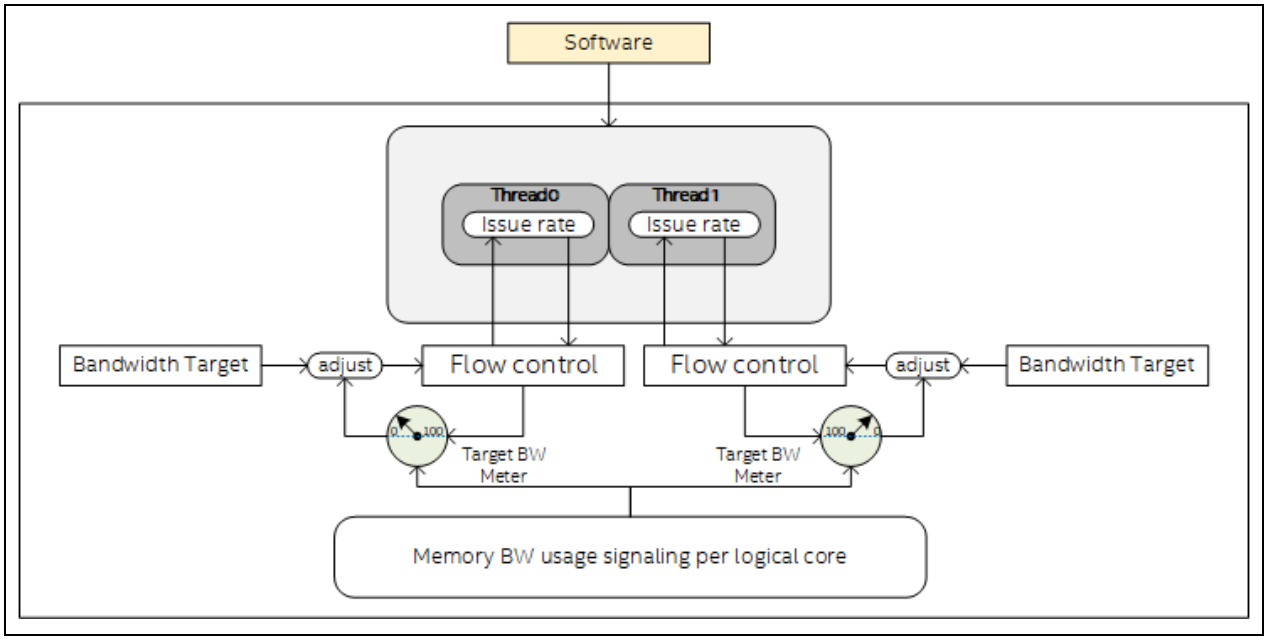
The third-generation MBA feature on future processors based on the codename Granite Rapids microarchitecture further enhances the feature with per-logical-processor control and a further improved controller design. Total memory bandwidth (all L3 miss traffic) is now managed by MBA.

This implementation follows the past MBA precedent of delivering significant enhancements without a major software overhaul, and while preserving backward compatibility.

3.2.4.3.1 Hardware Changes

The third generation of MBA builds upon the hardware controller introduced in the previous generation, which enabled significant system-level benefits, while providing the new capability to independently throttle logical processors, rather than more coarse-grained per-core bandwidth control in prior generations. Bandwidth control values are no longer selected as the “min” or “max” of the two throttling values for the threads running on the core; instead, throttling values are independently and directly applied to each logical processor. The third generation MBA implementation is shown in Figure 3-8.

Figure 3-8. High-Level Overview of the Third Generation MBA Feature



While this enhancement means that more direct bandwidth control (throttling) of threads is possible, re-tuning of software may be helpful to comprehend the effects of Intel® Hyper-Threading Technology contention versus cache and memory contention, and the effects on software performance.

3.2.4.3.2 Software-Visible Changes

In order to allow software to change its tuning behavior and detect that per-logical-processor bandwidth control is supported on a particular product generation, a CPUID bit is added to the MBA CPUID leaf to indicate support. See "CPUID—CPU Identification" in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B for details.

Despite another significant improvement of the hardware controller infrastructure architecture and improved capabilities, controller responsiveness, new internal microarchitecture, and transient-arresting capabilities, no new software interface changes are required to make use of the third generation of MBA relative to prior generations. Software previously using the second-generation MBA min/max selection capability should discontinue the use of the MBA_CFG MSR. The third-generation MBA capabilities are the default mode of operation on the codename Granite Rapids server microarchitecture.

Note that the MBA MSRs are listed in Appendix A.5 for completeness, but details of these legacy MSRs are available in Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. See Appendix A.2 for MBA feature supported product details.

3.2.4.4 Region Aware Memory Bandwidth Allocation

Region Aware Memory Bandwidth Allocation (MBA) for CPU Agents extends the existing third generation (per-thread bandwidth control) MBA capabilities to include Region Aware bandwidth controls per RDT Class of Service (CLOS). The Memory Region definitions used for Region Aware MBM and MBA are shared across the features, as specified in the ACPI MRRM table, allowing simultaneous and consistent monitoring and allocation of memory bandwidth.

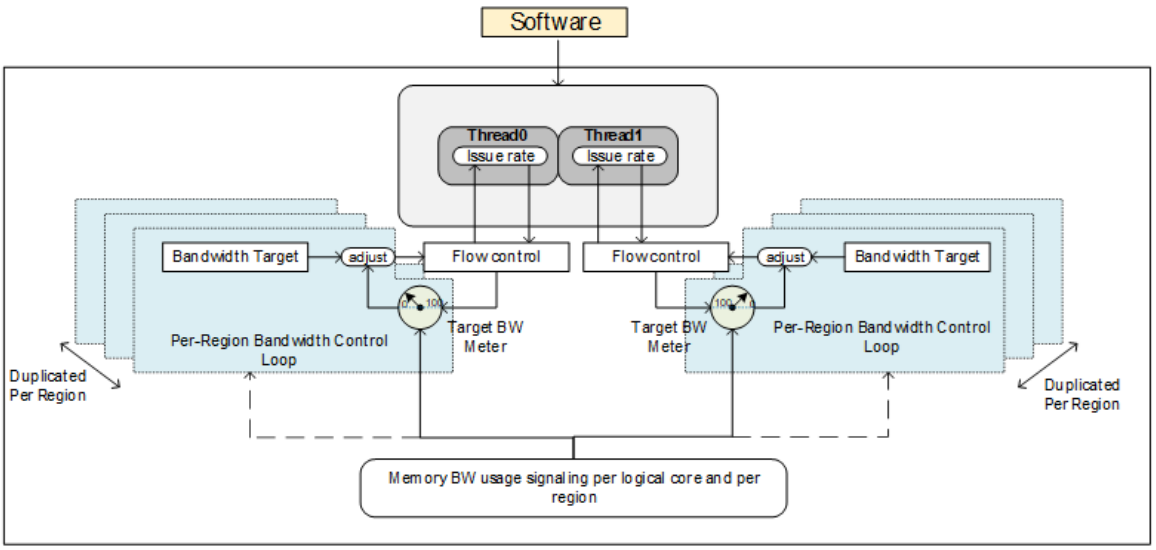
With Region Aware MBA, independent bandwidth control (throttling) of per-CLOS bandwidth to multiple regions is supported, allowing software to dynamically rebalance bandwidth control limits across different Memory Regions, which may have varying bandwidth, latency and capacity characteristics. Example uses include rebalancing bandwidth between VMs of different priority across a shared coherent interprocessor interconnect under the direction of a software control loop, or rebalancing bandwidth for threads of varying priorities across DRAM or CXL-backed memories. See Figure 3-9 for a high-level overview of Region Aware MBA and the following sections for details.

3.2.4.4.1 Region Aware MBA Overview

Region Aware MBA allows per-thread, per-CLOS, and per-Region control of Bandwidth to different regions — that is, enabling bandwidth control per-thread and per region simultaneously. As in the third generation of MBA, each region and thread are managed by a hardware controller which modulates the bandwidth of each thread targeting particular downstream region around the bandwidth targets set through the Intel RDT software interfaces.

The maximum number of regions is enumerable in the MRRM ACPI table described in Chapter 5, *BIOS Considerations*. The high-level implementation of Region Aware MBA is shown in Figure 3-9.

Figure 3-9. High-Level Overview of the Region Aware MBA



Unlike prior generations of MBA, the Region Aware MBA feature primarily uses an MMIO-based interface.

3.2.4.4.2 Enable MMIO Register

Region Aware MBA is enabled by software via an MMIO configuration register, before configuring per-thread, per-CLOS, and per-Region MBA bandwidth control values.

The RDT_CTRL MMIO register is used to control Region Aware MBA for CPU agents. The definition of the RDT_CTRL register is shown in Section 6.1.3.1. This register is scoped at the level of each Resource Management Domain, defined in Chapter 5. It is expected that software will configure this register consistently across all L3 caches present in the SoC.

The default value of the RDT_CTRL register is 0x4 (Region Aware MBA is disabled by default).

3.2.4.4.3 Min, Max and Optimal Bandwidth Caps per CLOS

The Region Aware MBA feature introduces three types of bandwidth limits per CLOS. Software may specify a minimum, maximum and optimal level of bandwidth target per CLOS. The specified range allows hardware to dynamically and autonomously manage bandwidth within the limits with fast response to changing system conditions or application phases while maximizing system throughput.

Bandwidth settings are described as follows:

- Maximum Cap: Caps the maximum bandwidth for a CLOS and any threads running in that CLOS. Allows the CLOS to switch to being constrained by a Max BW cap (which is typically above the Optimal level specified) if the resource is underutilized (utilized at a level less than the a medium or optimal rate).
- Optimal Cap: The software-preferred bandwidth control level for a given CLOS.
- Min Cap: Allows the hardware to attempt to guarantee a minimum amount of available bandwidth for a CLOS. Hardware may enforce this Lower BW cap below the Optimal level specified if the resource is over-utilized (typically at higher system region bandwidth utilization than medium but less than an overload scenario). Minimums are not necessarily guaranteed by hardware, as the sum of software requested minimums for instance may exceed the bandwidth that hardware can provide; as such, Min should be regarded as a best-effort minimum under heavy system utilization levels.

The Optimal Cap should be programmed to be between or equal to the Min and Max Cap levels. The Maximum Cap should be programmed to be greater than or equal to the Minimum and Optimal caps. Undesirable and undefined performance effects may result if cap programming guidelines are not followed.

The default hardware state is initialized with Max BW Cap == Optimal BW == Min BW Cap (and throttling is disabled by default).

Bandwidth control values in the MMIO-based feature interface (Chapter 6) are specified from unthrottled (maximum value "Q") down to a value of one (minimum bandwidth available, equivalent to maximum throttling). Bandwidth control values are implementation-specific and may not have an effect if more bandwidth is allowed than the processor is able to generate. Bandwidths may vary depending on traffic types, for instance various mixes of read and write traffic.

System software should consult the MARC sub-structure of the ERDT ACPI table to discover platform support for these caps as described in Chapter 5.

Software may choose to implement combined MBM and MBA control loops per-region to manage memory bandwidth of a set of processors, for instance comprising a virtual machine, to shape the bandwidth available to achieve goals such as prioritization or fairness.

3.2.5 Cache Bandwidth Allocation

Cache Bandwidth Allocation (CBA) allows an Operating System, Hypervisor, or similar system management agent to control internal core and correspondingly the downstream memory bandwidth for each of the logical processors. This feature is complimentary to MBA and provides OS/VMMs with the ability to throttle threads within the core.

The CBA feature along with the existing MBA feature provides a system-wide mechanism to throttle the bandwidth across different caches in the system including external memory, as well as control within a processor core or module. In combination, CBA and MBA provide both deterministic control and dynamic management of bandwidth resources to meet system Service Level Objectives (SLOs). The CBA feature reuses and extends existing constructs from the Intel RDT feature set including Classes of Service (CLOS).

A given CLOS used for L3 CAT for instance means the same thing as a CLOS used for CBA. Infrastructure such as the MSR used to associate a thread with a CLOS (the IA32_PQR_ASSOC_MSR) and some elements of the CPUID enumeration (such as CPUID leaf 10H (Cache Allocation Technology Enumeration Leaf)) are shared.

The Cache Bandwidth Allocation (CBA) feature provides control over bandwidth available between Level 1 (L1) caches, Level 2 (L2) Caches, and Level 3 (L3) Caches (as applicable) for each of the logical processors. Since reducing upstream bandwidth can also reduce bandwidth to external memory, this also provides an indirect control of memory bandwidth. This indirect control of external memory bandwidth can also reduce memory bandwidth. The CBA feature along with the MBA provides a mechanism to control the bandwidth of different applications.

Software should understand that the effective throttling applied to an application may be the maximum of the two values requested through the CBA

and MBA bandwidth control interfaces (the maximum resolved amount of throttling will be applied).

Similar to Intel RDT features, CBA includes the following key ingredients:

- A mechanism to enumerate the CBA capability to control the bandwidth from each level of the cache (for example, L1, L2, L3) to the next level (CPUID).
- A mechanism for the OS or Hypervisor to configure the amount of bandwidth available to a logical processor with a particular Class of Service via a throttle Level (MSRs, discussed later).
- Mechanisms for the OS or Hypervisor to signal the Class of Service to which an application belongs (the PQR MSR).
- Hardware mechanisms to guide and enforce the bandwidth throttle level across the cache hierarchy.

In some usages, the software may measure the memory bandwidth consumed by a given thread, application, VM or container at different Levels of cache hierarchy and external memory using performance monitor events and Memory Bandwidth Monitoring (MBM). Once the memory bandwidth is measured software can dynamically adjust the bandwidth control (throttling) level for the Class of Service (CLOS) used by that application. In other usages, software control loops may monitor application performance and adjust bandwidth control levels dynamically to achieve certain performance targets.

Certain processors, including those without an L3 cache, may implement the CBA feature without the presence of MBA. Other processors may choose to implement MBA, CBA or both.

More specific feature details about CBA are provided in the Intel® Architecture Instruction Set Extensions and Future Features. Note that the MSRs are listed in Appendix A.5. See Appendix A.2 for CBA feature supported product details.

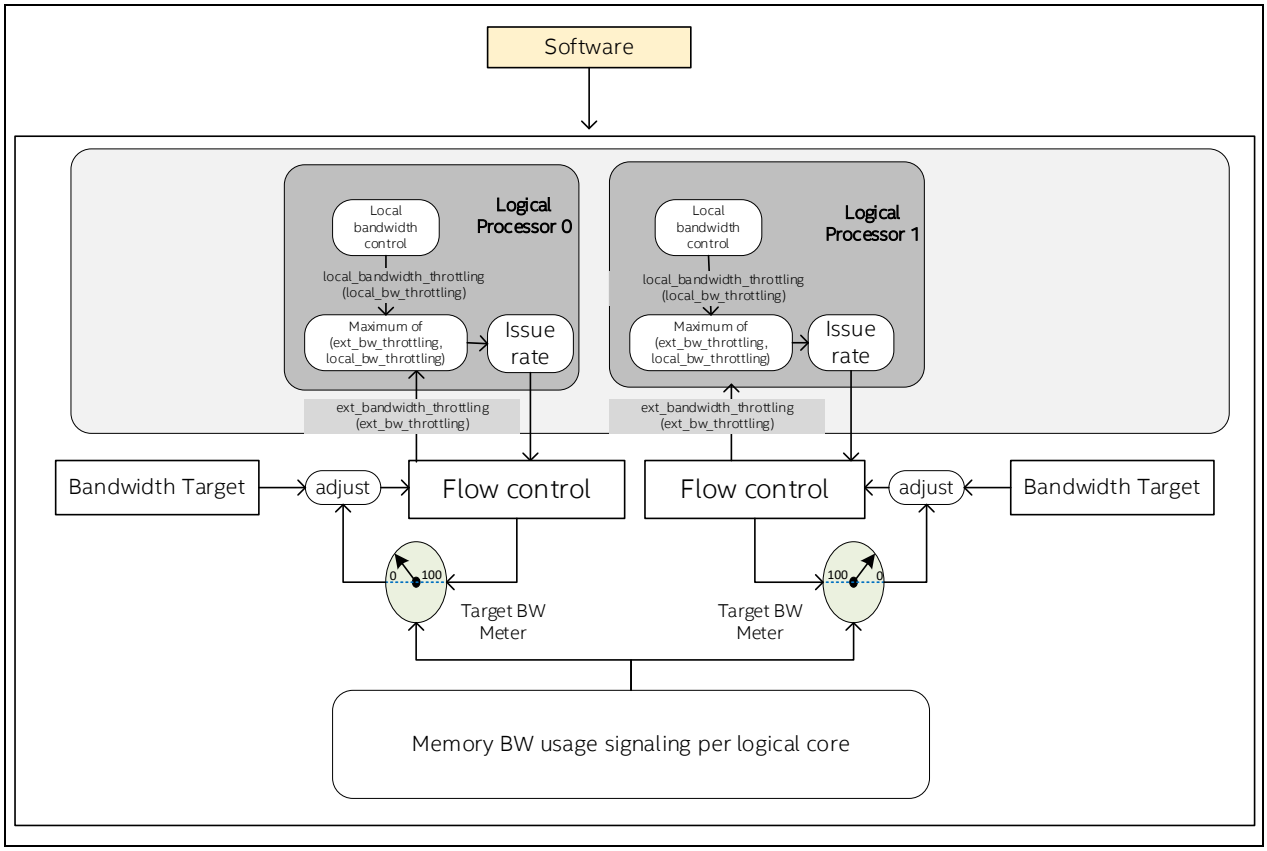
3.2.5.1 CBA Overview

The CBA feature implements a local hardware controller which when enabled provides the capability to independently throttle memory bandwidth of the logical processors across cache hierarchy and complements the MBA controller which throttles the external memory bandwidth.

3.2.5.2 Example of CBA Bandwidth Control Mechanism

An example of the bandwidth control enforced between the L2 cache and L3 cache is the maximum of the bandwidth throttling from the local CBA controller within the logical processor and the MBA hardware controller. An example CBA implementation is shown in Figure 3-10.

Figure 3-10. Example of CBA Bandwidth Control between L2 and L3 caches



3.2.5.3 Software Interface

In order to allow software to adapt its tuning behavior and detect that Cache Bandwidth Allocation is supported on a particular product generation, a CPUID bit is added to the Intel RDT Allocation CPUID leaf to indicate support (details are provided in the Intel® Architecture Instruction Set Extensions Manual).

The IA32_PQR_ASSOC MSR specifies the Class of Service associated with each logical processor. The CBA feature defines a set of MSRs known as IA32_QoS_Core_BW_Thrtl_n which provide a byte-encoded field for each CLOS to program the memory bandwidth throttle level. A higher value of throttling level means more bandwidth throttling and lower number indicates lesser throttling. The CPUID of the CBA feature enumerates the number of levels and maximum level supported by the logical processor. The reset value of each of the CLOS throttle values of the logical processor is 0 which indicates unthrottled bandwidth (zero throttling).

Each of the fields in the CBA IA32_QoS_Core_BW_Thrtl_n MSRs may be programmed from 0 to the maximum throttle level provided in the CPUID. If a value beyond the range from 0 to maximum throttle level is programmed, it will cause a #GP(0) fault. The Resource Management Domain (RMD) for CBA is per logical processor and thus the IA32_QoS_Core_BW_Thrl_n MSRs are logical

processor scope. Further details are provided in the Intel® Architecture Instruction Set Extensions and Future Features Programming reference manual.

3.2.5.4 Software Usage

The next sequence of steps provides a typical software usage of CBA feature:

1. System is setup with the desired workloads.
2. The software can use the performance counters along with MBM counters when available to profile and understand the bandwidth characteristics of the application.
3. The system administrator sets up the bandwidth control (throttling) level field in the IA32_QoS_Core_BW_Thrtl_n MSR (for example, in the VMM) to enforce the desired limits and the CLOS for each application. They can also monitor the bandwidth to confirm the setting is appropriate and adjust when needed.

In some cases, a specialized application software such as in embedded or communications usages will be able to communicate the memory bandwidth and latency requirements. This information may be used be performance management software to program the RDT features including CBA to meet the software memory bandwidth and latency requirements.

§

4 Intel® Resource Director Technology for Non-CPU Agents

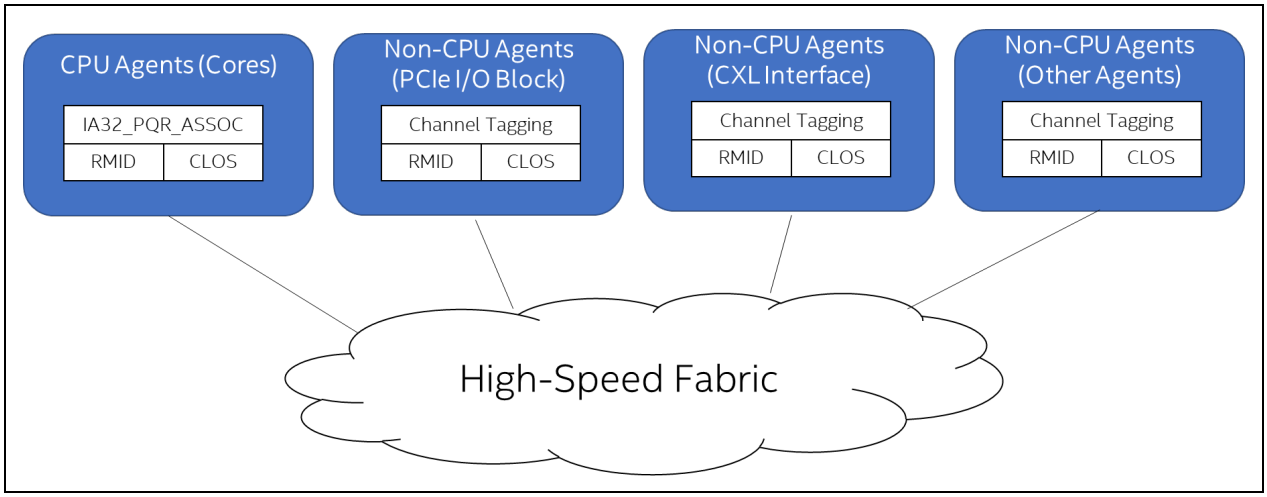
This chapter details Intel RDT features for non-CPU agents. Discussion is included on use cases and how Intel RDT monitoring, and controls are provided for non-CPU agents through extensions to the foundational CPU Agent Intel RDT features. Chapter 3 describes the components of the Intel RDT feature set which are common.

4.1 Introduction

Intel RDT for non-CPU agents comprises a set of features to monitor and control the resource utilization of non-CPU agents including PCI Express* (PCIe*) [5] and Compute Express Link (CXL)* [6] devices and integrated accelerators. The feature set enables monitoring usage of shared cache and memory bandwidth and control of cache usage by non-CPU agents. This feature set provides the equivalent CPU agent Intel RDT capabilities of CMT, MBM, and CAT for I/O devices.

The non-CPU agent Intel RDT includes controls at the device level and channel-level granularity in some cases. However, this granularity is fundamentally coarser than for software threads. CPU cores may execute hundreds of threads, all of which are tagged with RMIDs and CLOS, whereas an I/O device such as a NIC may serve hundreds of software threads, but it may only be monitored and controlled at a device level or channel level (see subsequent sections for details on channel-level monitoring and controls.)

Figure 4-1. Non-CPU Agent Building Atop CPU Agent Intel® RDT Features



4.2 Features

Cache Monitoring Technology (CMT) provides visibility into the cache (typically L3 cache). CMT provides occupancy counters on a per-RMID basis for non-CPU agents so cache occupancy (for example, capacity used by a particular RMID for I/O agents) can be tracked and read back dynamically during system operation. See Appendix A.2 for L3 CMT feature supported product details.

L3 Total and Local External Memory Bandwidth Monitoring (MBM) allows system software to monitor the usage of bandwidth between L3 cache and local or remote memory by non-CPU agents on a per-RMID basis. See Appendix A.2 for L3 Total and Local External MBM feature supported product details.

Cache Allocation Technology (CAT) allows control over shared cache capacity on a per-CLOS basis for non-CPU agents, enabling both isolation and overlap for better throughput, fairness, determinism and differentiation. See Appendix A.2 for L3 CAT feature supported product details.

4.3 Enumeration

Intel RDT uses the CPUID instruction to enumerate supported features and uses architectural Model-Specific Registers (MSRs) as interfaces to the monitoring and allocation features as described in Chapter 3 and in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.

There are no CPUID leaves or sub-leaves that are created for non-CPU agent Intel RDT; rather, existing CPUID leaves are augmented. CPUID.0xF(Shared Resource Monitoring Enumeration leaf).[ResID=1]:EAX[bit 9,10] enumerates presence of CMT and MBM features for non-CPU agents. CPUID.0x10(Cache Allocation Technology Enumeration Leaf).[ResID=1(L3 CAT)]:ECX[bit 1] enumerates the presence of the L3 CAT feature for non-CPU agents. Refer to Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, for CPUID details.

Additional enumeration information for Intel RDT for non-CPU agents is provided in the I/O Intel RDT table (IRDT), a vendor-specific extension to Advanced Configuration and Power Interface (ACPI) [4]. The IRDT table provides information on supported features, the structure of devices attached to particular links behind I/O blocks, the forms of tagging and controls supported on each link, and the specific MMIO interfaces used to control a given device. Details of IRDT are described in Chapter 5.

Confirming the presence of Intel RDT for CPU agents is a prerequisite for using the equivalent non-CPU agent Intel RDT feature. A compatibility matrix is provided in Appendix A.4. If a particular CPU agent Intel RDT feature is not present, any attempt to use non-CPU agent Intel RDT equivalents will result in a general protection fault in the MSR interface. Attempts to enable unsupported features in the I/O complex will result in writes to the corresponding MMIO enable or configuration interfaces being ignored.

Software may use the existing CPUID leaves to gather the maximum number of RMID and CLOS tags for each resource level (for example, L3 cache), and non-CPU agent Intel RDT is also subject to these limits.

Some platforms may support a mix of features, for instance supporting L3 CAT and the non-CPU agent Intel RDT equivalent, but no CMT or MBM monitoring.

4.4 Interface

Before configuring non-CPU agent Intel RDT (through MMIO), the feature should be enabled. The presence of one or more CPUID bits indicating support for one or more non-CPU agent Intel RDT features implies the presence of the IA32_L3_IO_RDT_CFG architectural MSR. This MSR is used to enable the non-CPU agent Intel RDT features.

Two bits are defined in this MSR. IRAE (Bit[0]) enables non-CPU agent RDT resource allocation features. IRME (Bit[1]) enables non-CPU agent RDT monitoring features.

The non-CPU agent Intel RDT Monitoring bit is supported if CPUID indicates that one or more non-CPU agent Intel RDT resource monitoring features are present.

The non-CPU agent Intel RDT Allocation bit is supported if CPUID indicates that one or more non-CPU agent Intel RDT resource allocation features are present.

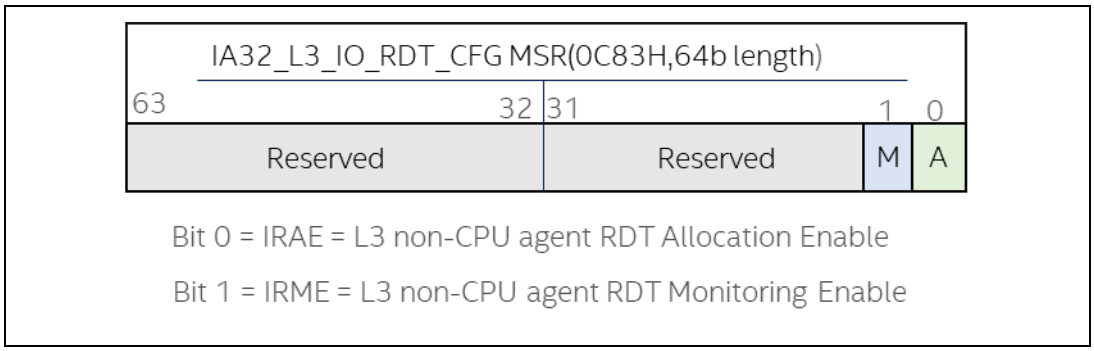
The default value is 0x0 (both the monitoring and allocation features are disabled by default). All bits not defined are reserved. Any writes to reserved bits will generate a General Protection Fault (#GP(0)).

This MSR is die-scoped and is cleared on system reset. It is expected that software will configure this MSR consistently across all L3 caches that may be present on a particular SOC die.

The definition of the IA32_L3_IO_RDT_CFG MSR is shown in Figure 4-2, and its MSR address is 0C83h.

Non-CPU agent RDT uses the RMID and CLOS tags in the same way that they are used for CPU agents.

Figure 4-2. The IA32_L3_IO_QOS_CFG MSR for Enabling Non-CPU Agent Intel® RDT



MMIO interfaces, discussed in subsequent sections, are defined by non-CPU agent Intel RDT to enable devices and/or channels to be tagged with RMIDs and CLOS, as applicable.

An example of device tagging with RMIDs, and CLOS is shown in Figure 4-3, where a PCIe device and a CXL device are tagged for monitoring and control of upstream resources in the L3 cache (shown within the fabric). Note that CPU cores are also shown, and as defined in the CPU agent Intel RDT feature set, their bandwidths may be controlled with the Memory Bandwidth Allocation (MBA) feature set.

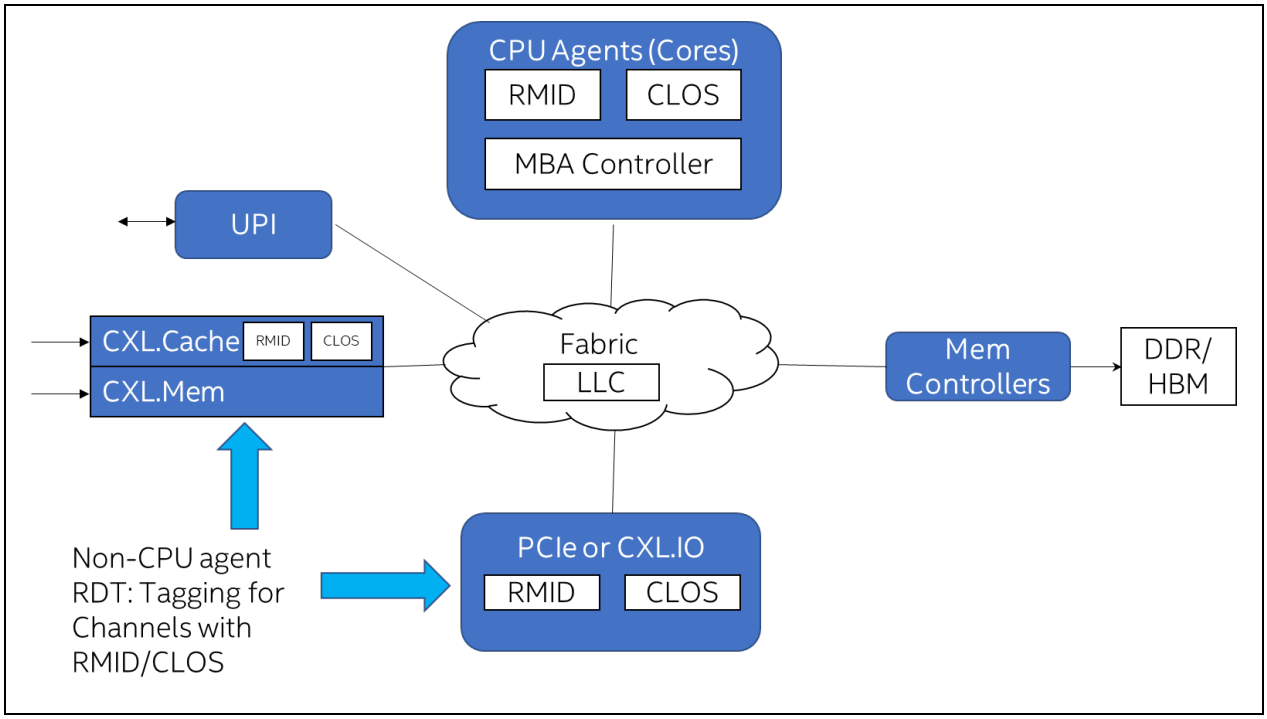
In the model of Figure 4-3, cores, PCIe devices and CXL devices are symmetrically arranged about the fabric and are symmetric in their ability to use RMIDs and CLOS.

The Intel RDT monitoring data retrieval MSRs IA32_QM_EVTSEL and IA32_QM_CTR are used for monitoring usage by non-CPU agents in the same way that they are used for Intel RDT for CPU agents for shared-L3 configurations. In certain configurations, memory-mapped registers may be provided to enable Intel RDT monitoring data retrieval for non-CPU agents. These memory-mapped registers are enumerated via ERDT ACPI (see section 5.1).

The CPU cache capacity control MSR interfaces are also used for controlling I/O device access to the L3 cache. The CLOS assigned to the device and the corresponding capacity bitmask in the IA32_L3_QOS_MASK_n MSR governs the fraction of the L3 cache into which the data may be filled, as described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. Certain I/O data flows may be implemented in processors with a multiple-L3 configuration, which may use memory-mapped register interfaces enumerated via ERDT ACPI (see section 5.1) for cache capacity control.

The CLOS tag retains the same meaning with regard to L3 fills for both CPU agents and non-CPU agents. Other cache levels may also be applicable depending on model-specific data flow patterns, which are governed by how I/O device data is filled into the cache in a model-specific fashion as governed by a given product generation's implementation of the DDIO (the Data Direct I/O feature).

Figure 4-3. Tagging for PCIe and CXL Devices



4.5 Common Tags

Non-CPU agent Intel RDT allows the traffic and operation of non-CPU agents to be associated with RMIDs and CLOS. In CPU agent Intel RDT, RMIDs and CLOS are numeric tags which may be associated with the operation of a thread through the IA32_PQR_ASSOC MSR. In non-CPU agent Intel RDT, a series of MMIO interfaces may be used to associate I/O devices with RMID and CLOS tags, and the numerical interpretation of the tags remains the same.

To wit, a particular CLOS tag, such as CLOS[5], means the same thing from the perspective of an CPU core or a non-CPU agent, and the same holds for RMIDs. In this fashion, RMIDs and CLOS used for non-CPU agents are said to be drawn from a "common pool" of RMID or CLOS tags, defined at the common L3 configuration level. Often these tags have specific meanings at a particular level of resource such as the L3 cache.

With non-CPU agent Intel RDT, specific devices may be selected for monitoring and control, and software enumeration and control are added to (1) enable non-CPU agent Intel RDT to build atop CPU agent Intel RDT, and (2) to comprehend the topology of devices behind I/O links, such as PCIe or CXL, and (3) to enable association of devices with RMID and CLOS tags.

4.6 I/O Blocks and Channels

I/O interfacing blocks are used to bridge from the ordered, non-coherent domain (such as PCIe) to the unordered, coherent domain (for example, the

shared interconnect fabric hosting the L3 cache). The non-CPU agent Intel RDT interface describes the devices connected behind each I/O complex (which may contain downstream PCIe root ports or CXL links) and enables configuration RMID/CLOS tagging for the same.

The I/O architecture is formalized as shown next. Channel mapping may occur anywhere between the device and the I/O block.

Figure 4-4. Mapping of Channels in the I/O Domain (PCIe Example)

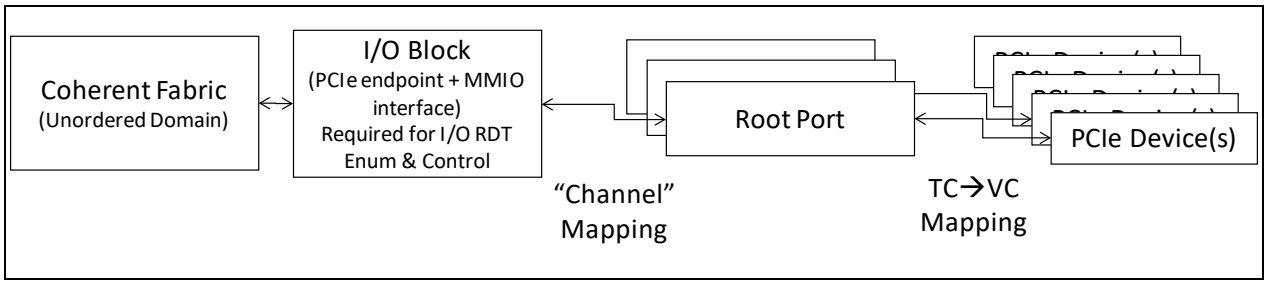
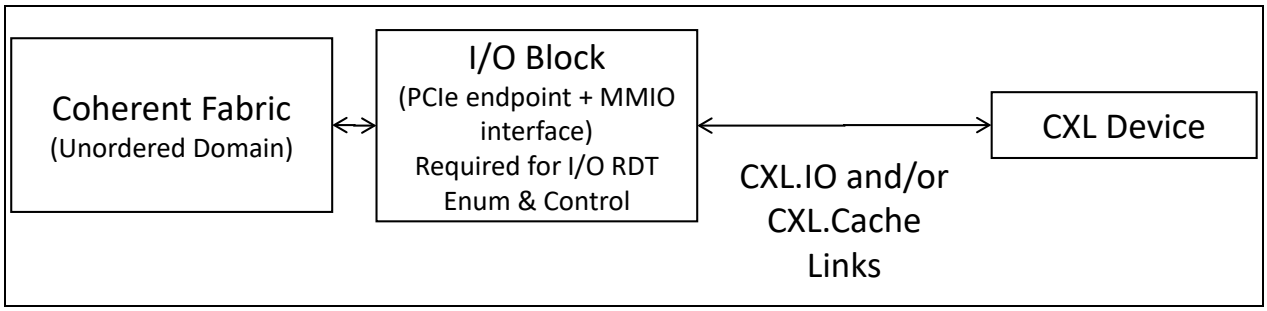


Figure 4-5. Mapping of Channels in the I/O Domain (CXL Example)



As shown in Figure 4-4, PCIe devices connected through a root port are routed through an I/O block, which applies non-CPU agent Intel RDT tagging (RMID and CLOS tagging) before traffic reaches the coherent fabric. Device traffic which is routed on various TCs and mapped to VCs, as defined in the PCIe specification [5], may be mapped to internal "Channels" between the root port and the I/O block. The non-CPU agent Intel RDT enumeration structures define the mapping between PCIe VCs and the non-CPU agent Intel RDT Channels so that software may perform tagging configuration based on Channels for platforms which support this capability (see the following sections for more detail).

An example with CXL [6] is shown in Figure 4-5. In this case a CXL.IO and CXL.Cache link may be in use, and the I/O block is again responsible for tagging, if supported. The links (CXL.IO and CXL.Cache) are controlled separately, through separate software interfaces. (See Chapter 7 for MMIO control interfaces.)

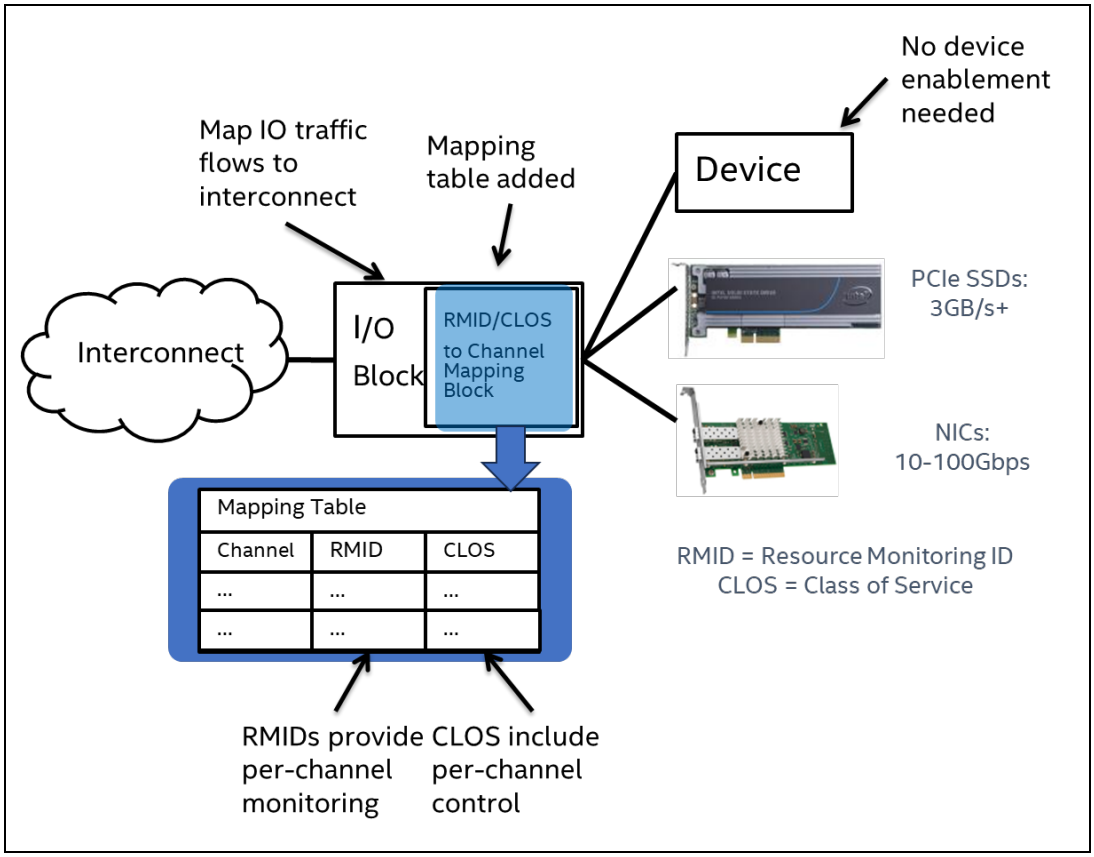
4.7 I/O Block Configuration

As described in the preceding section, PCIe devices mapped through their VCs to "Channels" may be configured on a per-Channel basis in the I/O Block. CXL

is a subset example of this, with the same configuration format, but only one configuration entry (the equivalent of a single Channel).

An enumerated number of Channels are supported in IRDT ACPI and configured through an MMIO interface to a "Mapping Table", as shown in Figure 4-6. A number of downstream PCIe devices may be mapped to various channels, and their traffic streams may be tagged, as applicable, through configuration of the I/O block.

Figure 4-6. Resource Monitoring and Control for PCIe and CXL Endpoints



4.8 Shared-L3 Configuration

The following sub-sections describe shared-L3 configuration and non-CPU agent Intel RDT features interplay.

4.8.1 Software Flow

Key software actions required to utilize non-CPU agent Intel RDT include (1) enumeration of the supported capabilities and details of that support, and (2) usage of the features through architectural platform interfaces.

- The software may enumerate the presence of non-CPU agent Intel RDT through a combination of parsing bit fields from CPUID and the IRDT ACPI table. The CPUID infrastructure provides basic information on the level of

CPU agent Intel RDT and non-CPU agent Intel RDT support present, and details of the common CLOS/RMID tags shared with CPU agent Intel RDT. The IRDT ACPI extensions provide many more details on non-CPU agent RDT specifically, such as which I/O blocks support non-CPU agent Intel RDT and where the control interfaces to configure the I/O blocks are located in MMIO space.

- Once software has enumerated the presence of non-CPU agent Intel RDT, configuration changes may be made through selecting a subset of RMID/CLOS tags to use with non-CPU agent Intel RDT, and configuring resource limits for those tags through MSRs for shared platform resources such as L3 cache (for example, for I/O use of L3 CAT) may be configured through the I/O block MMIO interfaces (the location of which is enumerated via IRDT ACPI).
- After resource limits are associated, RMID/CLOS tagging may be applied to the I/O device upstream traffic by assigning each I/O device into RMID/CLOS tags through its mapping to channels (and corresponding configuration through the MMIO interfaces for each I/O block, the location of which is enumerated via IRDT ACPI).
- It should be noted that while upstream shared SoC resources like L3 cache are monitored and controlled via shared RMID/CLOS tags, certain resources which are closer to the I/O may be controlled locally within each I/O block. In this view, RMIDs and CLOS are used for upstream resources which may be shared with CPU cores, but capabilities unique to the I/O device domain are controlled through I/O block-specific interfaces.
- Once tags are assigned and resource limits are applied, upstream traffic from I/O devices, though I/O blocks are tagged with the corresponding RMIDs/CLOS and such traffic is monitored and controlled within the shared resources of the SoC, much as CPU agent resources are controlled against these tags in CPU agent Intel RDT.
- As the IRDT ACPI tables used to enumerate non-CPU agent Intel RDT are generated by the BIOS, in the event of a hot-plug operation the OS or VMM software should update its internal tracking of device mappings based on newly added or removed device.
- In the case of bifurcation of a set of PCIe lanes, downstream devices which may be mapped to individual Channels may still be separately tagged and controlled, but devices sharing Channels will be mapped together against the same RMID/CLOS tags. As CXL devices have no notion of Channels, in the case of a bifurcated CXL link all downstream devices will be subject to the same RMID/CLOS.

4.8.2 Monitoring: Data Flows for RMIDs

As previously described, once RMID tags are applied to non-CPU agent traffic, all RMID-driven counter infrastructure in the platform may be used with non-CPU agent Intel RDT. In the case of the features in Appendix A.2 for instance, RMID-based cache occupancy and memory bandwidth overflow data is collected for non-CPU agents and may be retrieved by software. For each supported Cache Monitoring resource type, hardware supports only a finite

number of RMIDs. CPUID.(EAX=0FH(Shared Resource Monitoring Enumeration leaf), ECX=1H). ECX enumerates the highest RMID value that can be monitored with this resource type, see the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B for details.

As the interfaces for CPU agent Intel RDT data retrieval for RMID-based counters area already defined, the same interfaces are used, including MSR-based data retrieval for the corresponding set of three Event IDs (EvtIDs) defined for CPU agent Intel RDT's CMT and MBM features (See Chapter 3).

RMIDs are allocated to devices by software from the pool of RMIDs defined at the L3 cache level, and the IA32_QM_EVTSEL / IA32_QM_CTR MSRs can be used to specify RMIDs and Event IDs and retrieve data.

The MSR pair used to retrieve event data is shown in Figure 3-3, however as all properties are inherited from CPU agent RDT (See Chapter 3 for details). All of access rules and usage sequence, reserved bit properties, initial values, and virtualization properties are inherited from CPU agent Intel RDT.

4.8.3 Allocation: CLOS-based Control Interfaces

The Intel RDT Allocation features for non-CPU agent use CLOS-based tagging for control of cache at a given level, subject to where data fills from I/O devices in a particular cache and SoC implementation. In common cases this will be the last-level cache (L3) as described in the ACPI – specifically in the IRDT sub-table known as RCS and its flags. Software may adjust the levels of cache that it controls based on the expected level(s) of cache into which I/O data may fill subject to flags in the RCS. This in turn may affect which CPU agent CAT control masks software programs to control the data fills of non-CPU agents and may vary depending on how a particular RCS is connected to shared resources on a platform.

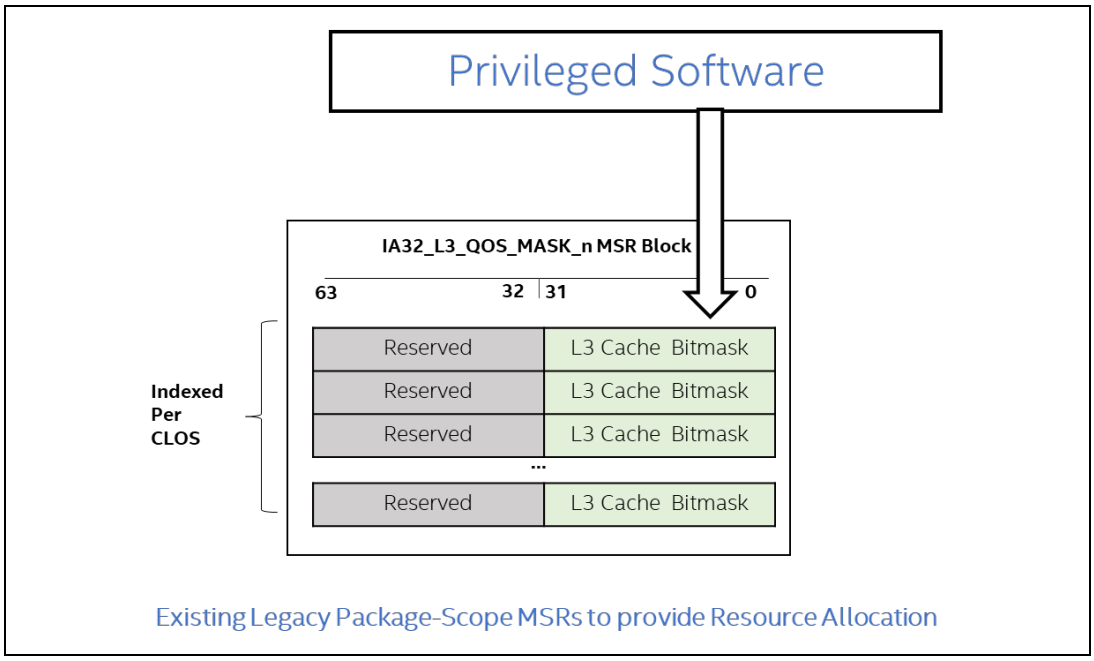
For each supported Cache Allocation resource type, the hardware supports only a finite number of CLOS. CPUID.(EAX=10H(Cache Allocation Technology Enumeration Leaf), ECX=2):EDX[15:0] reports the maximum CLOS supported for the resource (CLOS are zero-referenced, meaning a reported value of "15" would indicate 16 total supported CLOS). Bits 31:16 are reserved, see the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B for details.

In a typical example, with a non-CPU agent (for example, a PCIe device) filling data into an L3 cache, the RCS structure's "Cache Level Bit Vector" would have bit 17 set to indicate the L3 cache, and software may control the CPU agent Intel RDT L3 CAT masks (in IA32_L3_QoS_MASK_n MSRs) to define the amount of cache into which non-CPU agents may fill. As with RMID management, the CLOS used in this context are drawn from the pool at the applicable resource (L3 cache in this context).

If other cache levels are introduced or used in the future, incremental software enabling may be required to comprehend fills into other cache levels.

As the masks used for control are drawn from the existing definitions of such cache controls in the CPU agent Intel RDT definitions, details such as reserved fields, initialization values, and so on, are defined in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B. Figure 4-7 shows an example of the CPU agent Intel RDT L3 CAT control MSRs.

Figure 4-7. Reuse of the IA32_L3_QOS_MASK_n MSRs for L3 CAT Control



4.9 CXL-Specific Considerations

This section describes CXL-specific device considerations including management of traffic on multiple links and CXL device types.

4.9.1 CXL block Interfacing Fundamentals

CXL devices may connect to an RMUD via multiple RCSes, and independent control of each RCS may be required. See Chapter 5 for RMUD and RCS details.

Non-CPU agent Intel RDT features provide monitoring and controls for CXL.IO and CXL.Cache link types. CXL.mem is not subject to controls in the I/O block as it is viewed as a resource rather than an agent in Intel RDT terms. Instead bandwidth to CXL.mem is controlled at the agent source (for example, using MBA) as previously described and where supported.

4.9.2 Integrated Accelerators

Integrated accelerators, including those using integrated CXL links, may be monitored and controlled using the semantics described in preceding sections.

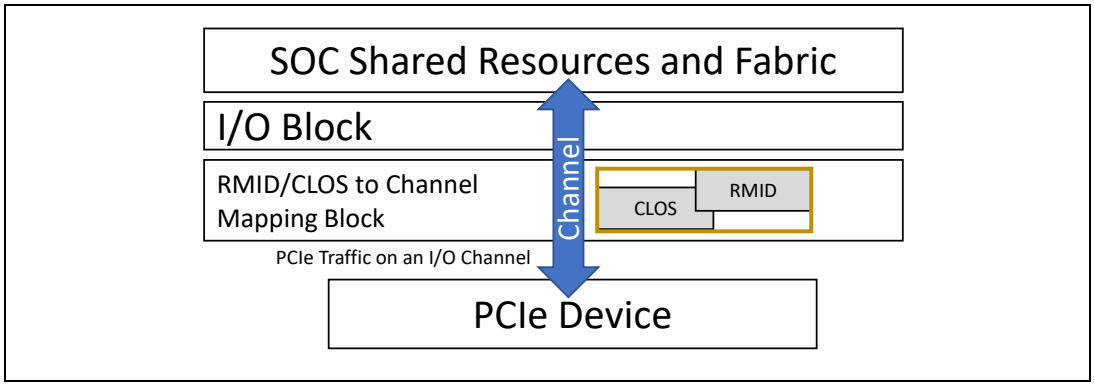
4.10 Use Cases

A number of non-CPU agent Intel RDT use cases are described in this section involving PCIe, CXL, and integrated accelerators.

As an implementation of the architectural model shown in Figure 4-4 and Figure 4-5, I/O block tags upstream DMA traffic (such as PCIe writes) as shown in Figure 4-8, enabling the device's resource utilization in the shared resources of the fabric, such as L3 cache, to be monitored and controlled through Intel RDT RMIDs and CLOS.

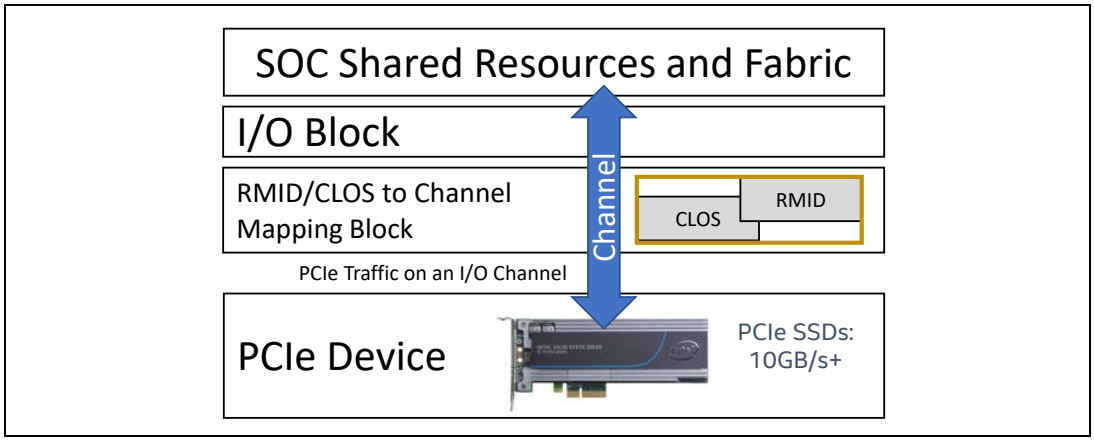
The applicable features for each tag are described in Appendix A.2, and software may configure these tags as described in Chapter 5, which describes the ACPI; see the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, for CPUID enumeration, and Section 4.8 and Chapter 7 for how the software may actuate these controls.

Figure 4-8. Device Traffic Tagging Model with PCIe as the Sole Traffic Path



As a concrete example, Figure 4-9 shows a high-performance PCIe SSD, subject to tagging with CLOS (so that its L3 cache footprint may be controlled), and RMIDs (so that its L3 cache occupancy and overflow bandwidth to memory may be monitored).

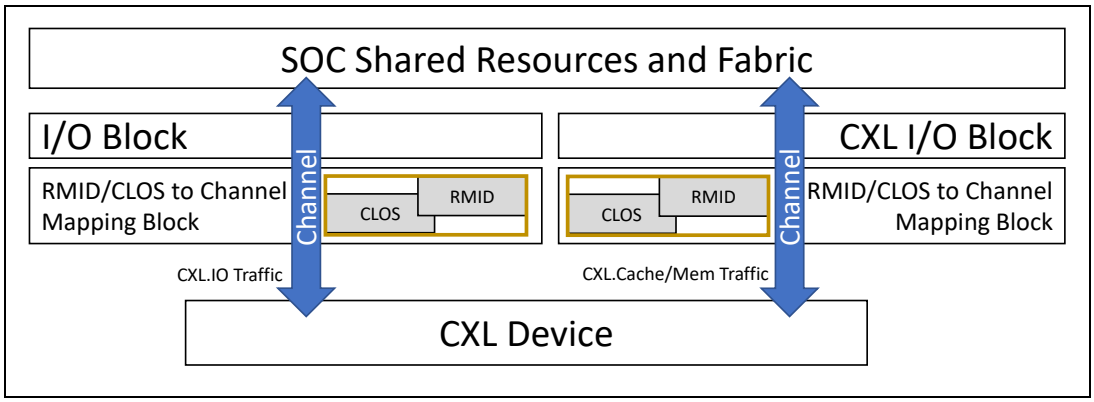
Figure 4-9. PCIe Device Example, with Traffic on a Channel Tagged with an RMID and CLOS



An example with a CXL device is shown in Figure 4-10, in which two paths are used for the device's traffic, one over CXL.IO, and one over CXL.Cache, through two separate I/O blocks, and note that the CXL.Cache link defines only one channel. In such a case, the software may configure RMID and CLOS tagging separately for the links. The links operate independently.

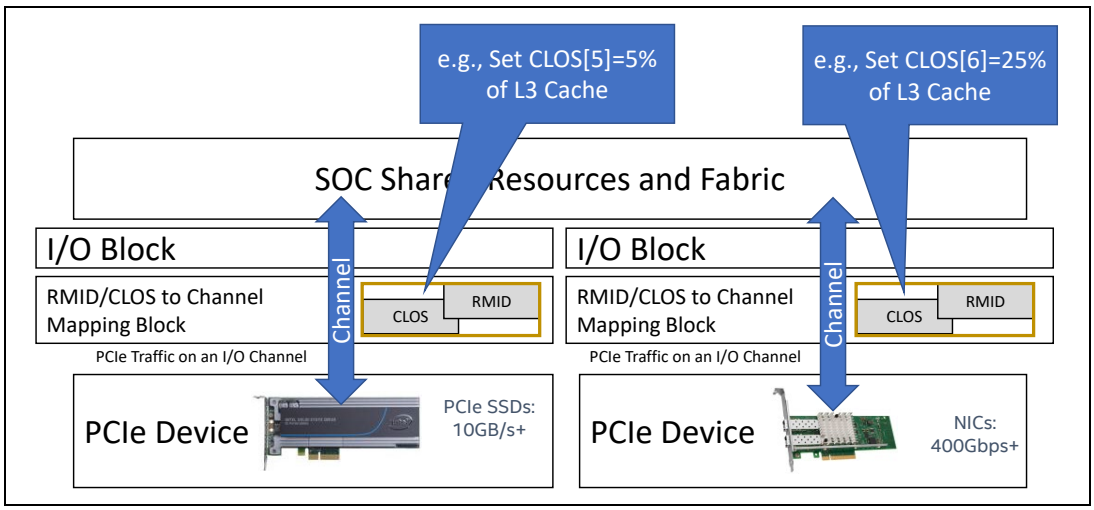
Note that no controls are provided for CXL.Mem, as the use of CXL.Mem resolves around accessing memory on a target device, and bandwidths from logical processors may be controlled with Intel RDT's Memory Bandwidth Allocation (MBA) feature. A more detailed discussion of this case surrounds Figure 4-14.

Figure 4-10. CXL Example of Device Tagging Model with CXL.IO and CXL.Cache Traffic Paths



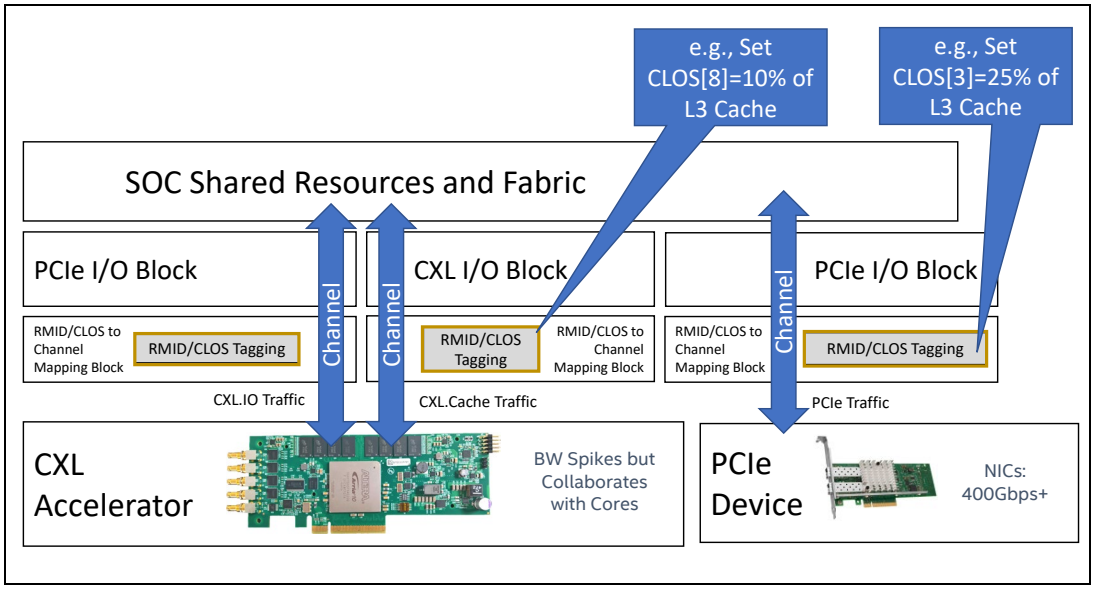
An example with multiple devices with different properties is shown in Figure 4-11, where a pair of PCIe devices on separate I/O blocks may be controlled independently, with separate RMID and CLOS tags. In this case a PCIe SSD which does not utilize the cache effectively may be limited, but a NIC which fills into the cache for data to be consumed by CPU cores may be prioritized.

Figure 4-11. Example of Controlling Two Different PCIe Devices



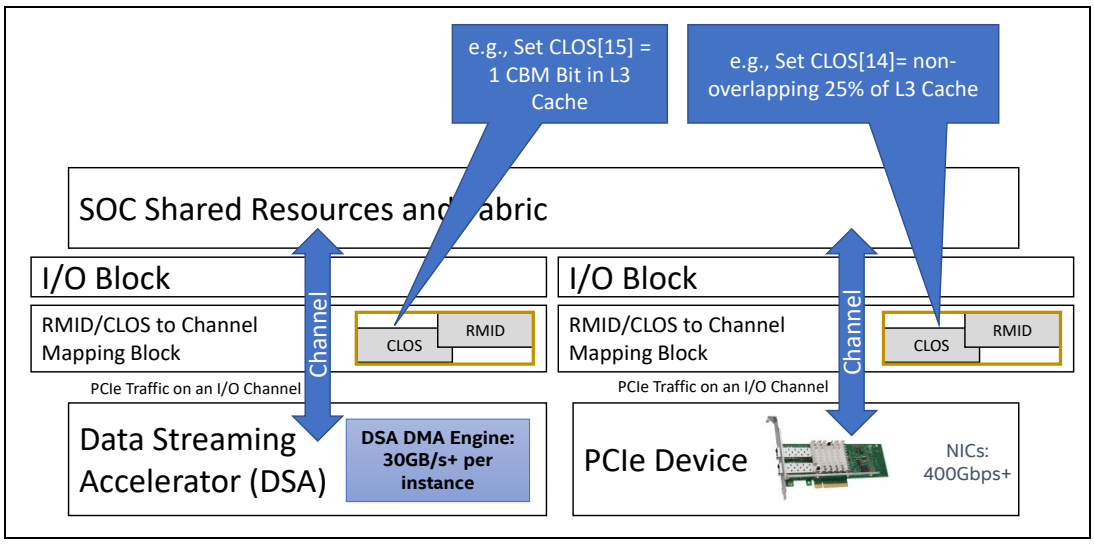
The following image shows an example with one CXL accelerator, perhaps a CXL-enabled FPGA card, utilizing CXL.IO and CXL.Cache, controlled independently from an I/O block with a PCIe device attached.

Figure 4-12. Example of Controlling a CXL Accelerator



An example of tagging and controlling an integrated accelerator, the Data Streaming Accelerator (DSA) alongside a PCIe device is shown in Figure 4-13. Depending on system load conditions and the DSA usage case, software may choose to allocate non-overlapping portions of the cache to minimize cache contention effects.

Figure 4-13. Example of Controlling a High-Bandwidth Integrated Accelerator

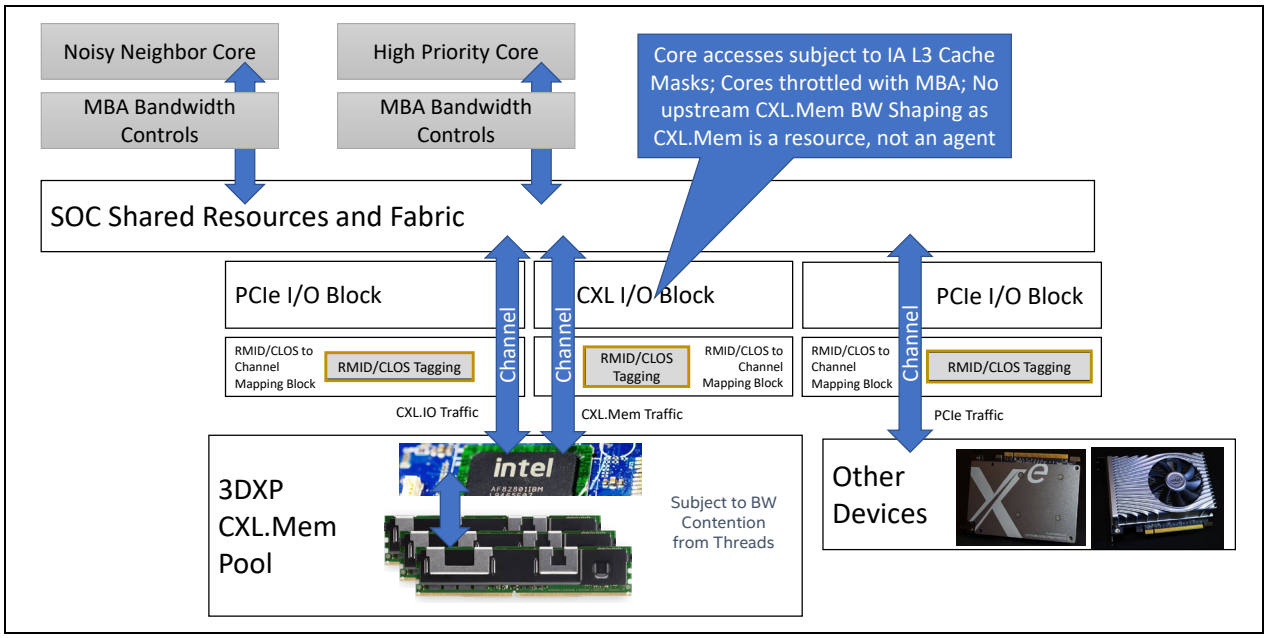


A complex example with multiple features in use is shown in Figure 4-14, where various PCIe devices are controlled with non-CPU agent Intel RDT, but a CXL device is also present, using CXL.IO and CXL.Mem links. The CXL device may be tagged and controlled on its CXL.IO interface.

As the main purpose of CXL.Mem is for host accesses to device memory, however, traffic responses up through the CXL.mem path are not subject to MBA bandwidth shaping, though they are sent with RMID and CLOS tags. If bandwidth is constrained on this link and software seeks to redistribute bandwidth across different priorities of accessing agents, such as CPU cores, the MBA feature may be used to redistribute bandwidth and throttle at the source of the requests (the agent's traffic injection point).

This example shows that for comprehensive management of cache and bandwidth resources on the platform, a combination of CPU agent Intel RDT and non-CPU agent Intel RDT controls may be necessary.

Figure 4-14. MBA to Control a CXL.Mem Pooling Device



§

5 BIOS Considerations

Software may query processor support of RDT shared resource monitoring and allocation features by executing CPUID for the RDT features which are defined per CPU Agent (independent of interface scope). ACPI structures may then be consulted for further details on the Enhanced RDT (ERDT structure) features support, memory range-to-region mapping (MRRM structure) and I/O RDT RMID/CLOS (IRDT structure) topologies and tagging. These ACPI tables enumerate the location of specific MMIO interfaces used to allocate or monitor shared platform resources. All numeric values in ACPI-defined tables, blocks, and structures are always encoded in little endian format. Signature values are stored as fixed-length strings.

5.1 Introduction to Enhanced RDT Interfaces

Two new structures are defined to enumerate the Region Aware MBM and MBA features and the shared infrastructure such as memory regions that they use:

1. **Enhanced RDT (ERDT) ACPI structure**: Describes the resource management domains (RMDs) in an SoC and which agents are managed within the scope of each resource management domain; this structure also describes the architectural MMIO register locations for various resource monitoring and allocation features.
2. **Memory Range and Region Mapping (MRRM) ACPI structure**: Describes distinct memory ranges in the platform along with their Region-ID mapping registers, in order to group *ranges* into *regions* for Region-Aware Memory Bandwidth Allocation (MBA) and Memory Bandwidth Monitoring (MBM). This structure may be used by other Intel product features which utilize or reference the same consistent Region-IDs.

Features defined in ERDT are dependent on the regions defined within the MRRM table. If ERDT is defined but not MRRM, software may assume that only one memory region is defined, covering all system memory. If MRRM is defined but not ERDT, software may assume that no region-aware RDT features are present.

5.2 ERDT Table Structure Layout

The top-level ACPI structure defined to support Enhanced RDT features is the "ERDT" structure. Figure 5-1 exemplifies the ERDT ACPI hierarchy. As described in the following sections, the ERDT structure may include the following defined sub-structures:

- Resource Management Domain Description Structure (RMDDs),
- CPU Agent Collection Description Structure (CACDs),
- Device Agent Collection Description Structure (DACDs),

- Cache Monitoring Registers for CPU Agents Description Structure (CMRCs),
- Memory Bandwidth Monitoring Registers for CPU Agents Description Structure (MMRCs),
- Memory Bandwidth Allocation Registers for CPU Agents Description Structure (MARCs),
- Cache Allocation Registers for CPU Agents Description Structure (CARCs),
- Cache Monitoring Registers for Device Agents Description Structure (CMRDs),
- I/O Bandwidth Monitoring Registers for Device Agents Description Structure (IBRDs),
- Cache Allocation Registers for Device Agents Description Structure (CARDs),
- I/O bandwidth Allocation Registers for Device Agents Description Structure (IBADs)

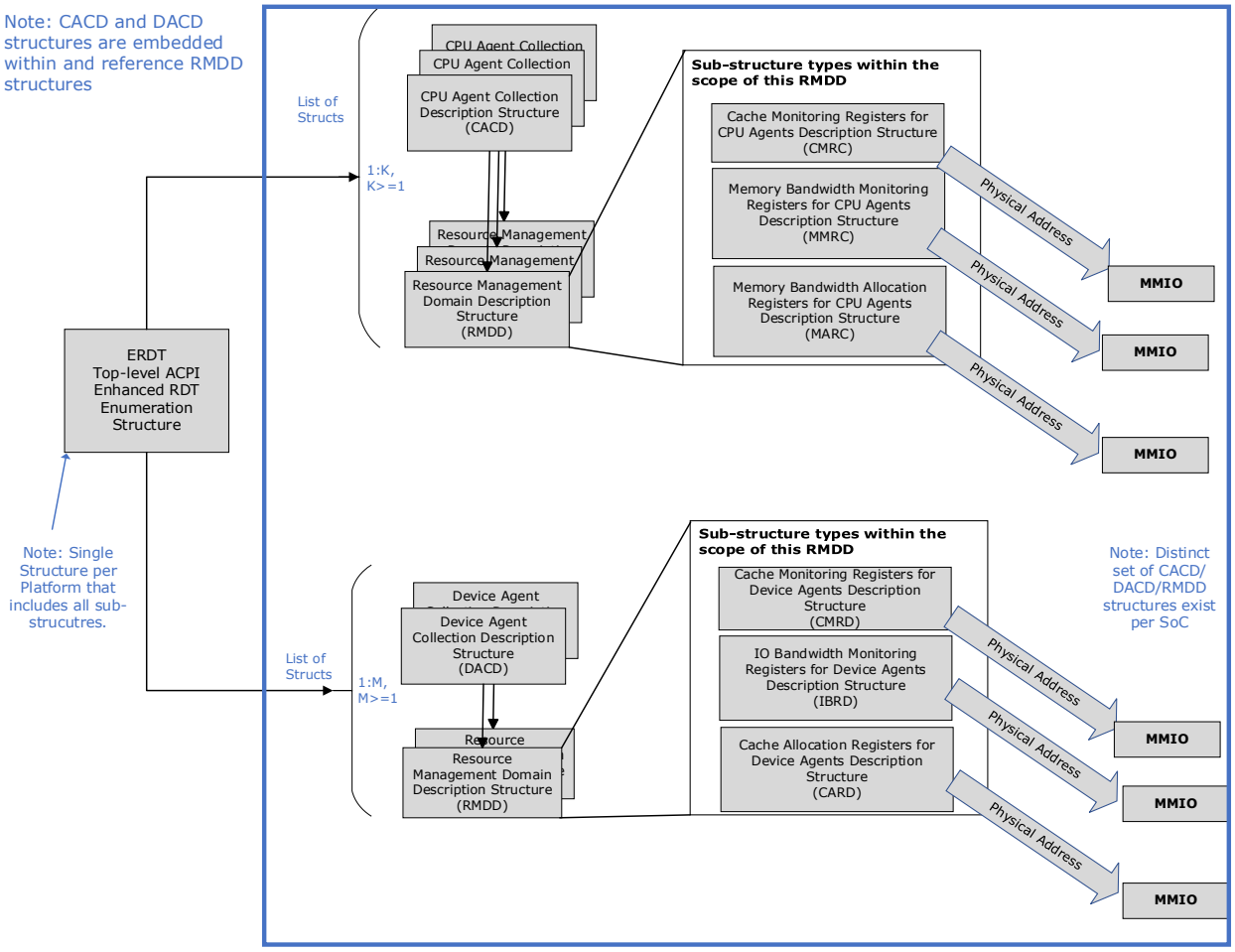
There exists only one instance of the ERDT table for a given platform. Each RMDD structure within ERDT represents a resource management domain (RMD). Thus, there will be as many RMDDs as the number of resource management domains across all SoCs on the platform. For example, on a dual-socket platform, where each socket hosts N resource management domains, there will be 2*N RMDD sub-structures within ERDT.

CPU agents under the scope of each resource management domain (RMDD) are enumerated (via their x2APIC physical APIC-ID [1]) through a CPU Agent Collection Description (CACD) table. Similarly, non-CPU agents under the scope of an RMDD are enumerated through a Device Agent Collection Description (DACD) table. Each RMDD table has a unique Domain-ID, and the CACD/DACD table instances correlate to the corresponding RMDD by referencing the respective RMDD Domain-ID value.

As shown in Figure 5-1, the CMRC, MMRC and MARC sub-structures describe the architectural MMIO register location and organization for the Cache Monitoring Technology (CMT), Memory Bandwidth Monitoring (MBM) and Memory Bandwidth Allocation (MBA) enhanced features in RMDDs which have CPU agents within scope. Similarly, the CMRD, CARD and IBRD registers describe the architectural MMIO register locations and organization for I/O CMT, I/O CAT and I/O MBM registers in RMDDs with non-CPU agents within scope. As feature support may differ across RMDDs, software should individually enumerate all ERDT sub-structures to determine whether asymmetric feature support is present.

See Section 5.4 for complete details about these structures.

Figure 5-1. Top-level Structure of ERDT ACPI Enumeration

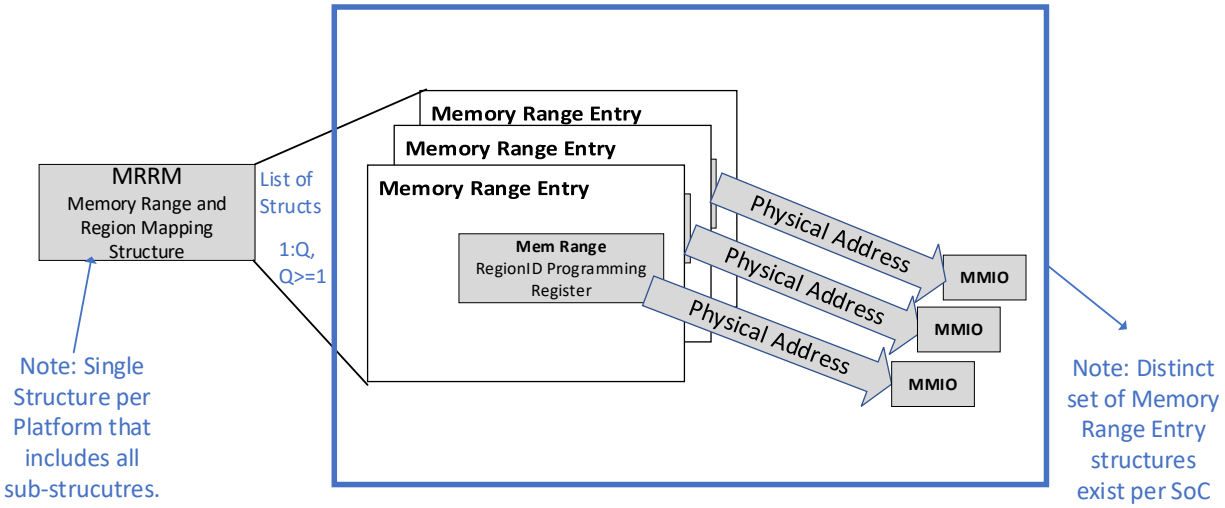


5.3 MRRM Table Structure Layout

Figure 5-2 shows the MRRM ACPI table structure which describes the memory range to region mapping details. Each memory range entry in the MRRM structure consists of a contiguous range of host physical address (HPA) space along with the registers (if hardware and OS configuration of Region-IDs are supported) for programming Region-ID for this memory range. Each memory range may be configured with a Region-ID for local accesses and a Region-ID for remote (cross-socket) accesses. The memory ranges are identical to the memory ranges specified in the Memory Affinity Structure specified in the ACPI SRAT structure [8] and may be cross-referenced by address as described in a later chapter.

If the platform supports only static memory range to region mapping (as with initial implementations), then the 'Platform-assigned Static Local Region-ID' and 'Platform-assigned Static Remote Region-ID' fields (Section 5.5.1) describe local and remote Region-IDs allocated by platform firmware (BIOS) for that memory range.

Figure 5-2. Top-level Structure of MRRM ACPI Enumeration



5.4 ERDT Table Structure Details

5.4.1 ERDT Structure Format and Field Descriptions

The top-level ACPI table, known as the Enhanced Resource Director Technology Structure (ERDT) is shown below. This table includes a unique signature, and its enumerated length includes all sub-structures. The length of the ERDT table is variable.

Table 5-1. Enhanced Resource Director Technology (ERDT) Top-Level ACPI Structure

Field	Byte Length	Byte Offset	Description
Signature	4	0	"ERDT". Signature for the Enhanced Resource Director Technology Description structure.
Length	4	4	Length, in bytes, of the description table including the length of the associated sub-structures.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID
OEM Table ID	8	16	For the ERDT structure, the Table ID is the manufacturer model ID
OEM Revision	4	24	OEM Revision of the ERDT Table for OEM Table ID.
Creator ID	4	28	Vendor ID of utility that created the table.
Creator Revision	4	32	Revision of utility that created the table.

Field	Byte Length	Byte Offset	Description
Max CLOS	4	36	Maximum number of Classes Of Service (CLOS) supported by the platform for resource allocation management. The CLOS values supported by the platform are 0 through N, where N is the value reported in this field.
Reserved	24	40	Reserved (0).
ERDT Sub-structures	-	64	List of ERDT sub-structures. All sub-structures have a type and length fields at the beginning. The type field uniquely identifies the type of sub-structure, and the length field indicates the size of the sub-structure including the size of any subordinate structures it may include. For forward compatibility, software is expected to ignore and skip any sub-structures that it does not recognize. The following table lists the various sub-structures defined.

5.4.2 Valid ERDT Sub-structure Types

All RDT Sub-structures start with a 'Type' field (two bytes) followed by a 'Length' field (two bytes) indicating the size in bytes of the structure (including sub-structures).

Table 5-2. Valid ERDT Sub-structure Types

Type	Abbreviation	Description
0	RMDD	Resource Management Domain Description Structure
1	CACD	CPU Agent Collection Description Structure
2	DACD	Device Agent Collection Description Structure
3	CMRC	Cache Monitoring Registers for CPU Agents Description Structure
4	MMRC	Memory-bandwidth Monitoring Registers for CPU Agents Description Structure
5	MARC	Memory-bandwidth Allocation Registers for CPU Agents Description Structure
6	CARC	Cache Allocation Registers for CPU Agents Description Structure
7	CMRD	Cache Monitoring Registers for Device Agents Description Structure
8	IBRD	IO Bandwidth Monitoring Registers for Device Agents Description Structure
9	IBAD	IO bandwidth Allocation Registers for Device Agents Description Structure
10	CARD	Cache Allocation Registers for Device Agents Description Structure

Type	Abbreviation	Description
>10		Reserved for future use. For forward compatibility, software skips structures it does not comprehend by skipping the number of bytes indicated by the Length field.

BIOS implementations should report these RDT Sub-structure types in numerical order, that is, all RDT sub-structures of Type 0 (RMDD) enumerated before remapping structures of Type 1 (CACD) and Type 2 (DACD). All of the valid sub-structures which are under the scope of Type 0 (RMDD) should be enumerated in numerical order, e.g., Type 1 (CACD), Type 2 (DACD), Type 3 (CMRC), Type 4 (MMRC) and so forth and then subsequent Type 0 (RMDD) enumeration should take place. Defined in this fashion, not all of these are top-level structures, some of these sub-structure types may exist under the scope of other structure type such as an RMDD. See Section 5.4.3.1 for details.

5.4.3 Resource Management Domain Description Structure

A Resource Management Domain Description (RMDD) structure describes a RDT resource management domain. There must be at least one instance of this structure present to represent one or more enhanced features such as CMT, MBM and MBA if supported.

Table 5-3. Resource Management Domain Description (RMDD) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	0 - Resource Management Domain Description (RMDD) structure.
Length	2	2	Total Length of this RMDD and all sub-structures within the scope of this RMDD.
Flags	2	4	**Bit 0: L3 Domain** • If Set, this RMDD represents a resource-management domain hosting a CPU L3 cache. The relevant registers are described through CMRC, MMRC, MARC and CARC register description structures. CPU L3 cache details are reported through CPUID. Please refer to the Intel SDM. **Bit 1: I/O L3 Domain** • If Set, this RMDD represents a resource-management domain hosting an L3 cache into which I/O data may fill. The relevant registers in this resource management domain are described through CMRD, IBRD and CARD register description structures. I/O L3 details are reported in 'Number of I/O LLC slices', 'Number of I/O LLC sets' and 'Number of I/O LLC ways' fields which may align with CPU caches but is not guaranteed. Cache line size is the same for I/O L3 and CPU caches, and is reported in CPUID.0x4.EBX[11:0]. Bits 2-15: Reserved.
Number of I/O L3 slices	2	6	This field is valid only if bit 1 (indicating I/O L3 domain) is set in the flags field. A

Field	Byte Length	Byte Offset	Description
			value of Q in this field indicates the number of slices forming this I/O L3 cache.
Number of I/O L3 sets	1	8	This field is valid only if bit 1 (indicating I/O L3 domain) is set in flags field. A value of N in this field indicates 2^N number of sets for the I/O L3 supported under this Resource Management Domain's scope.
Number of I/O L3 ways	1	9	This field is valid only if bit 1 (indicating I/O L3 domain) is set in flags field. A value of Q in this field indicates the number of I/O L3 ways supported under this Resource Management Domain's scope.
Reserved	8	10	Reserved(0)
DomainID	2	18	This field indicates a unique Domain ID for the RMDD structure representing this resource management domain. The CPU and device agents under the scope of an RMDD are enumerated through CACD and DACD structures referencing the value in this field.
Max RMID	4	20	Maximum **Resource Monitoring IDs (RMID)** number supported by this resource management domain. The value reported is specific to the respective domain. The RMID values supported are 0 through X, where X is the value reported in this field. Max RMID is only valid if monitoring sub-features are supported for this domain.
Control Register Base Address	8	24	4KB aligned host physical address of control registers for this RDT Domain.
Control Register Size	2	32	The size of the control register space for this domain, in units of 4 KB pages.
RMDD structures	-	34	A list of agent collection description structures and register description structures within the scope of this RMDD. All sub-structures have a type and length fields at the beginning. The type field uniquely identifies the type of sub-structure, and the length field indicates the size of the sub-structure including the size of any subordinate structures it may include. For forward compatibility, software is expected to ignore and skip any sub-structure types that it does not recognize. The Table 5-4 lists the various sub-structure types defined.

5.4.3.1 Valid Sub-structure Types within the scope of this RMDD

All RDT Sub-structures start with a 'Type' field (two bytes) followed by a 'Length' field (two bytes) indicating the size in bytes of the structure (including sub-structures).

Table 5-4. Valid Sub-structure Types within the scope of an RMDD

Type	Abbreviation	Description
1	CACD	CPU Agent Collection Description Structure
2	DACD	Device Agent Collection Description Structure
3	CMRC	Cache Monitoring Registers for CPU Agents Description Structure
4	MMRC	Memory-bandwidth Monitoring Registers for CPU Agents Description Structure
5	MARC	Memory-bandwidth Allocation Registers for CPU Agents Description Structure
6	CARC	Cache Allocation Registers for CPU Agents Description Structure
7	CMRD	Cache Monitoring Registers for Device Agents Description Structure
8	IBRD	IO Bandwidth Monitoring Registers for Device Agents Description Structure
9	IBAD	IO bandwidth Allocation Registers for Device Agents Description Structure
10	CARD	Cache Allocation Registers for Device Agents Description Structure
>10		Reserved for future use. For forward compatibility, software skips structures it does not comprehend by skipping the number of bytes indicated by the Length field.

Note that as described in Figure 5-1, ERDT may contain RMDD, CACD, DACD and other sub-structures. The CACD and DACD structures are embedded within and reference an RMDD.

BIOS implementations should report these sub-structure types in numerical order. i.e., All RDT substructures of Type 0 (RMDD) enumerated before remapping structures of Type 1 (CACD) and Type 2 (DACD). All the valid sub-structures which are under the scope of Type 0 (RMDD) should be enumerated in numerical order i.e., Type 1 (CACD), Type 2 (DACD), Type 3 (CMRC), Type 4 (MMRC) and so forth and then subsequent type 0 (RMDD) enumeration should take place.

5.4.4 CPU Agent Collection Description Structure

A CPU Agent Collection Description (CACD) structure uniquely represents a collection of logical processor agents on the platform managed by a common RDT domain. There must be at least one instance of this structure for each RDT domain supporting CPU agents.

Table 5-5. CPU Agent Collection Description (CACD) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	1 - CPU Agent Collection Description (CACD) Structure
Length	2	2	Varies (8 + size of Enumeration-IDs field)
Reserved	2	4	Reserved(0)

Field	Byte Length	Byte Offset	Description
RMDD DomainID	2	6	This field specifies the Domain-ID for the resource management domain that monitors/enforces cache and memory bandwidth resourcing for agents in this collection. Resource management domains are enumerated through the RMDD structures. Each RMDD structure includes a unique Domain-ID.
Enumeration-IDs []	-	8	Array of Enumeration-IDs, each representing a unique logical processor in this agent collection. Enumeration-ID of a logical processor is its 32-bit physical X2APIC ID as reported in the Processor Local x2APIC Affinity Structure in ACPI System Resource Affinity Table (SRAT).

5.4.5 Device Agent Collection Description Structure

A Device Agent Collection Description (DACD) structure uniquely represents a collection of device agents on the platform managed by a common RDT domain. There must be at least one instance of this structure for each RDT domain supporting devices.

Table 5-6. Device Agent Collection Description (DACD) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	2 - Device Agent Collection Description (DACD) Structure
Length	2	2	Varies (8 + size of Device Agent Scope Entries field)
Reserved	2	4	Reserved(0)
RMDD DomainID	2	6	This field specifies the Domain-ID for the resource management domain that monitors/enforces cache and memory bandwidth resourcing for agents in this collection. Resource management domains are enumerated through the RMDD structures. Each RMDD structure includes a unique Domain-ID.
Device Agent Scope Entries []	-	8	Array of one or more Device Agent Scope Entries that identify devices in this collection. Refer to Device Agent Scope Entry structure

5.4.5.1 Device Agent Scope Entry Structure

The Device Agent Structure is composed of Device Agent Scope Entries. Each Device Agent Scope Entry refers to either a PCI endpoint device or a PCI sub-hierarchy.

Table 5-7. Device Agent Scope Entry (DASE) Structure

Field	Byte Length	Byte Offset	Description
Type	1	0	The following values are defined for this field. **0x01: PCI Endpoint Device** - The device identified by the 'Path' field is a PCI endpoint device. **0x02: PCI Sub-hierarchy** - The device identified by the 'Path' field is a PCI-PCI bridge. In this case, the specified bridge device and all its downstream devices are included in the scope. Other values for this field are reserved for future use.
Length	1	1	Length of this Entry in Bytes. (6 + X), where X is the size in bytes of the "Path" field.
Segment Number	2	2	The PCI Segment associated with this device agent
Reserved	1	4	Reserved (0)
Start Bus Number	1	5	This field describes the bus number (bus number of the first PCI Bus produced by the PCI Host Bridge) under which the device agent identified by this Device Agent Scope Entry resides.
Path	2*N	6	For Device Agent Scope Entries with Type value of 0x1 or 0x2, this field describes the hierarchical path from the Host Bridge to the device specified by the Device Agent Scope Entry. For example, a device in a N-deep hierarchy is identified by N {PCI Device Number, PCI Function Number} pairs, where N is a positive integer. Even offsets contain the Device numbers, and odd offsets contain the Function numbers. The first {Device, Function} pair resides on the bus identified by the 'Start Bus Number' field. Each subsequent pair resides on the bus directly behind the bus of the device identified by the previous pair. The identity (Bus, Device, Function) of the target device is obtained by recursively walking down these N {Device, Function} pairs. If the 'Path' field length is 2 bytes (N=1), the Device Scope Entry identifies a 'Root-Complex Integrated Device'. The requester-id of 'Root-Complex Integrated Devices' are static and not impacted by system software bus rebalancing actions. If the 'Path' field length is more than 2 bytes (N > 1), the Device Scope Entry identifies a device behind one or more system software visible PCI-PCI bridges. Bus rebalancing actions by system software modifying bus assignments of the device's parent bridge impacts the bus number portion of device's requester-id.

5.4.6 Cache Monitoring Registers for CPU Agents Description Structure

A Cache Monitoring Registers for CPU Agents Description (CMRC) structure describes cache monitoring registers for CPU Agents in a RDT domain. There must be at least one instance of this structure for each RDT domain which includes a cache that supports occupancy monitoring. This structure is always contained within an RMDD structure.

Table 5-8. Cache Monitoring Registers for CPU Agents Description (CMRC) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	3 - Cache Monitoring Registers for CPU Agents Description Structure
Length	2	2	Fixed: 48B
Reserved	4	4	Reserved(0)
Flags	4	8	• **Bit 0: Unavailable Bit Support:** If Set, indicates CMT data registers in this domain support the Unavailable bit, signaling that data may be unavailable. If Clear, indicates CMT Register does not support the Unavailable bit field. See Section 6.1.3.2 for the CMT Register Layout. • Bits 1-31: Reserved.
Register Indexing Function Version	1	12	This field indicates Register Indexing Function Version Number. See 6.1.3.2 for details on the software usage guidance of this field.
Reserved	11	13	Reserved(0)
CMT Register Block Base Address for CPU	8	24	4KB aligned Host Physical Address of MMIO Registers used for RMID-granular near Cache Monitoring Technology for CPU agents.
CMT Register Block Size for CPU	4	32	Size of cache monitoring register space in units of number of 4KB pages. CMT registers are located in the range (X):(X+Y*4096), where X is the value reported in the Register Block Base Address field and Y is the value in this field. Refer to Section 6.1.3.2 for details on the cache monitoring register layout
CMT Register Clump Size for CPU	2	36	The registers in the Register Block are organized in “Clumps”. Each Register Clump is a set of N adjacent 8-Byte sized registers, where N is the value specified in this field. The size of a Register Clump is thus 8*N bytes.
CMT Register Clump Stride for CPU	2	38	The first Register Clump starts at the address specified by the base address field above. Each subsequent Register Clump starts at a fixed offset (stride) from the previous Register Clump. The Stride value (S) is reported as number of bytes in this field. Thus, registers in a given Clump 'C'

Field	Byte Length	Byte Offset	Description
			are located at byte offsets <C*S> to <C*S+8*N>
CMT Counter Upscaling Factor	8	40	Upscaling factor from reported CMT counter value to occupancy metric (bytes). See Intel® 64 Architecture Software Developer's Manual (SDM), Volume 3B, Chapter Title: Debug, Branch Profile, TSC, and Intel® Resource Director Technology (Intel® RDT) Features for details on upscaling Factor.

5.4.7 Memory Bandwidth Monitoring Registers for CPU Agents Description Structure

A Memory Bandwidth Monitoring Registers for CPU Agents (MMRC) Description structure describes memory bandwidth monitoring registers for CPU Agents in a RDT domain. There must be at least one instance of this structure for each RDT domain which supports monitoring of bandwidth to memory. This structure is always contained within an RMDD structure.

Table 5-9. Memory Bandwidth Monitoring Registers for CPU Agents Description (MMRC) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	4 - Memory-bandwidth Monitoring Registers for CPU Agents Description Structure
Length	2	2	Varies (56 + size of MBM Correction Factor field)
Reserved	4	4	Reserved(0)
Flags	4	8	• **Bit 0: Unavailable Bit Support:** If Set, indicates MBM data registers in this domain support the Unavailable bit, signaling that data may be unavailable. If Clear, indicates MBM Register does not support the Unavailable bit field. See Section 6.1.3.3 for the MBM Register Layout. • **Bit 1: Overflow Bit Support:** If Set, indicates MBM data registers in this domain support the Overflow bit. If Clear, indicates MBM data registers do not support the Overflow bit field. See Section 6.1.3.3 for discussion of MBM Register Layout and clear-on-read semantics. • Bits 2-31: Reserved.
Register Indexing Function Version	1	12	This field indicates Register Indexing Function Version Number. See Section 6.1.3.3 for details on the software usage guidance of this field.
Reserved	11	13	Reserved(0)
MBM Register Block Base Address	8	24	4KB aligned Host Physical Address of MMIO Registers used for RMID-granular Memory Bandwidth Monitoring (MBM)

Field	Byte Length	Byte Offset	Description
MBM Register Block Size	4	32	Size of Memory Bandwidth Monitoring register space in units of number of 4KB pages. MBM registers are located in the range (X):(X+Y*4096), where X is value reported in base address field and Y is the value in this field. Refer to Section 6.1.3.3 for details on the Memory Bandwidth monitoring register layout.
MBM Counter Width	1	36	A value Q indicates that Q-bit counter width is supported by underlying implementation.
MBM Counter Upscaling Factor	8	37	MBM data values read can be converted to bandwidth (in bytes) by multiplying with the Upscaling Factor.
Reserved	7	45	Reserved(0)
MBM Correction Factor List Length	4	52	A value in this field defines MBM Correction Factor List Length. Below are the valid values for MBM Correction List Length: **0:** Do not apply a correction factor to the MBM values. **1:** Apply a single correction factor specified in MBM Correction Factor field to all the MBM values (uniformly apply this correction factor to all data values retrieved from counters for all RMIDs). **Max RMID+1:** If the value in this field matches the maximum supported RMID + 1 for this domain (as RMIDs are zero-indexed), indicated in RMDD:"Max RMID", apply the indicated indexed correction factor specified in MBM Correction Factor list to the corresponding the RMID value for MBM counter.
MBM Correction Factor []	-	56	A list of MBM Correction Factors. The list will contain zero, one or Max RMID + 1 entries. Fixed point 32-bit format per entry in this list. Counter values may be multiplied by the correction factor to account for processor-specific implementation variations.

5.4.8 Memory Bandwidth Allocation Registers for CPU Agents Description Structure

A Memory Bandwidth Allocation Registers for CPU Agents Description (MARC) structure describes memory bandwidth allocation registers for CPU Agents in a RDT domain. There must be at least one instance of this structure for each RDT domain which supports Memory Bandwidth Allocation. This structure is always contained within an RMDD structure.

Table 5-10. Memory Bandwidth Allocation Registers for CPU Agents Description (MARC) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	5 - Memory-bandwidth Allocation Registers for CPU Agents Description Structure
Length	2	2	Fixed: 48B
Reserved	2	4	Reserved(0)
MBA Flags	2	6	MBA Control Window Parameter Flags: **Bit 0: MBA_OPTIMAL_CONTROL_WINDOW** ● If Set, this domain supports the Optimal BW Window control. ● If Clear, this domain does not support Optimal BW Control Window. **Bit 1: MBA_MINIMUM_CONTROL_WINDOW** ● If Set, this domain supports the Minimum BW Window control ● If Clear, this domain does not support Minimum Control Window. **Bit 2: MBA_MAXIMUM_CONTROL_WINDOW** ● If Set, this domain supports the Maximum BW Window control ● If Clear, this domain does not support Maximum BW Control Window. Bit 3-15 : Reserved (0)
Register Indexing Function Version	1	8	This field indicates Register Indexing Function Version Number. See Sections 6.1.3.4, 6.1.3.5, and 6.1.3.6 for details on the software usage guidance of this field.
Reserved	7	9	Reserved(0)
MBA Optimal BW Register Block Base Address	8	16	If the MBA_OPTIMAL_CONTROL_WINDOW flag is set, this field specifies the base 4KB-aligned Host Physical Address of the MMIO Registers used for Optimal Memory Bandwidth Allocation for each Class of Service
MBA Minimum BW Register Block Base Address	8	24	If MBA_MINIMUM_CONTROL_WINDOW flag is set, this field specifies the base 4KB-aligned Host Physical Address of the MMIO Registers used for Minimum Memory Bandwidth Allocation for each Class of Service
MBA Maximum BW Register Block Base Address	8	32	If MBA_MAXIMUM_CONTROL_WINDOW flag is set, this field specifies the base 4KB-aligned Host Physical Address of the MMIO Registers used for Maximum Memory Bandwidth Allocation for each Class of Service
MBA Register Block Size	4	40	Size of Memory Bandwidth Allocation registers in units of number of 4KB pages. A value of X in this field indicates X*4KB space for each of the optimal, minimum, and maximum register sets (if supported). Refer to Chapter 6 for details on the Memory Bandwidth Allocation register layout.

Field	Byte Length	Byte Offset	Description
MBA BW Control Window Range	4	44	A value of Q in this field indicates the permitted bandwidth control window range of values that can be programmed into MBA registers is 1 through Q, where a value of 1 represents maximum throttling and Q represents minimal throttling (maximum bandwidth).

More details on the programming and interpretation of the MBA BW Control Window Range field are provided in Section 3.2.4.4, *Region Aware Memory Bandwidth Allocation*, and in Section 6.1.3, Register Descriptions.

5.4.9 Cache Monitoring Registers for Device Agents Description Structure

A Cache Monitoring Registers for Device Agents Description (CMRD) structure describes near cache monitoring registers for Device Agents in a RDT domain. There must be at least one instance of this structure for each RDT domain which supports Cache Monitoring Technology (CMT). This structure is always contained within an RMDD structure.

Table 5-11. Cache Monitoring Registers for Device Agents Description (CMRD) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	7 - Cache Monitoring Registers for Device Agents Description Structure
Length	2	2	Fixed: 48B
Reserved	4	4	Reserved(0)
Flags	4	8	• **Bit 0: Unavailable Bit Support:** If Set, indicates CMT data registers in this domain support the Unavailable bit, signaling that data may be unavailable. If Clear, indicates CMT Register does not support the Unavailable bit field. See Section 6.1.3.7 for the CMT Register Layout. • Bits 1-31: Reserved.
Register Indexing Function Version	1	12	This field indicates Register Indexing Function Version Number. See Section 6.1.3.7 for details on the software usage guidance of this field.
Reserved	11	13	Reserved(0)
Register Base Address	8	24	Base address of the Device Agent register set for this CMRD. This address must be aligned according to the size of the register set size reported in the Register Block Size field of this structure.
Register Block Size	4	32	Size of register space in units of number of 4KB pages. Registers are located in the range (X):(X+Y*4096), where X is the value reported in the Register Block Base Address field and Y is the value in this

Field	Byte Length	Byte Offset	Description
			field. Refer to Section 6.1.3.7 for details on the register layout.
CMT Register Offset for I/O	2	36	Bits 0-11: This field specifies the offset to the CMT registers for I/O in its corresponding 4KB page. If the register base address is X, and the value reported in this field is Y, then the first address for the CMT register for I/O is calculated as (X+Y). Each subsequent CMT Register clump for I/O starts at the same CMT Register Offset for I/O in the next consecutive 4KB page. Bit 12-15: Reserved(0)
CMT Register Clump Size for I/O	2	38	The registers in the Register Block are organized in Clumps. Each Register Clump is a set of N adjacent 8-Byte sized registers, where N is the value specified in this field. The size of a Register Clump is thus 8*N bytes. Each Register Clump is organized in consecutive 4KB pages. Each Register clump starts at an offset specified by CMT Register Offset for I/O field in its corresponding 4KB page.
CMT Counter Upscaling Factor	8	40	Upscaling factor from reported CMT counter value to occupancy metric (bytes). See Intel® 64 Architecture Software Developer's Manual (SDM), Volume 3B, Chapter Title: Debug, Branch Profile, TSC, and Intel® Resource Director Technology (Intel® RDT) Features for details on upscaling Factor.

5.4.10 IO Bandwidth Monitoring Registers for Device Agents Description Structure

An IO Bandwidth Monitoring Registers for Device Agents Description (IBRD) structure describes total I/O BW and I/O Miss registers for Device Agents in a RDT domain. There must be at least one instance of this structure for each RDT domain which supports I/O Bandwidth Monitoring. This structure is always contained within an RMDD structure.

Table 5-12. IO Bandwidth Monitoring Registers for Device Agents Description (IBRD) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	8 - IO Bandwidth monitoring Registers for Device Agents Description Structure
Length	2	2	Varies (64 + size of I/O BW Correction Factor field)
Reserved	4	4	Reserved(0)

Field	Byte Length	Byte Offset	Description
Flags	4	8	• **Bit 0: Unavailable Bit Support:** If set, indicates IBRD counter registers support the Unavailable bit field. If clear, indicates that the IBRD Register does not support the Unavailable bit field. See Section 6.1.3.9 for IBRD Register Layout. • **Bit 1: Overflow Bit Support:** If set, indicates IBRD counter registers support the Overflow bit field. If clear, indicates that the IBRD Register does not support the Overflow bit field. See Section 6.1.3.9 for IBRD Register Layout. • Bits 2-31: Reserved.
Register Indexing Function Version	1	12	This field indicates Register Indexing Function Version Number. See Section 6.1.3.9 details on the software usage guidance of this field.
Reserved	11	13	Reserved(0)
Register Base Address	8	24	Base address of Device Agent register set for this IBRD. This address must be aligned according to the size of the register set size reported in the Register Block Size field of this structure.
Register Block Size	4	32	Size of register space in units of number of 4KB pages. Registers are located in the range (X):(X+Y*4096), where X is the value reported in the Register Block Base Address field and Y is the value in this field. Refer to Chapter 6 for details on the register layout.
Total I/O BW Register Offset	2	36	Bits 0-11: This field specifies the offset to the Total I/O BW registers in its corresponding 4KB page. If the register base address is X, and the value reported in this field is Y, the address for the Total I/O BW registers is calculated as (X+Y). Each subsequent Total I/O BW registers clump starts at the same Total I/O BW Register Offset in the next consecutive 4KB page. Bits 12-15: Reserved(0)
I/O Miss BW Register Offset	2	38	Bit 0-11: This field specifies the offset to the I/O Miss BW registers in its corresponding 4KB page. If the register base address is X, and the value reported in this field is Y, then the first address for the I/O Miss BW registers is calculated as (X+Y). Each subsequent I/O Miss BW registers starts at the same I/O Miss BW Register Offset in consecutive 4KB page. Bit 12-15: Reserved(0)
Total I/O BW Register Clump Size	2	40	The registers in the Register Block are organized in Clumps. Each Register Clump is a set of N adjacent 8-Byte sized registers, where N is the value specified in this field. The size of a Register Clump is thus 8*N bytes. Each Register Clump is organized in consecutive 4KB pages. Each Register

Field	Byte Length	Byte Offset	Description
			clump starts at offset specified by Total I/O BW Register Offset for I/O field in its corresponding 4KB page.
I/O Miss Register Clump Size	2	42	The registers in the Register Block are organized in "clumps". Each register clump is a set of N adjacent 8-Byte sized registers, where N is the value specified in this field. The size of a register clump is thus 8*N bytes. Each register clump is organized in consecutive 4KB pages. Each register clump starts at an offset specified by the Total I/O BW Register Offset for I/O field in its corresponding 4KB page.
Reserved	7	44	Reserved(0)
I/O BW Counter Width	1	51	A value Q indicates that Q-bit counter width is supported for Total I/O BW and I/O Miss BW counters by the underlying implementation.
I/O BW Counter Upscaling Factor	8	52	Total I/O BW and I/O Miss BW Counter value can be converted to bandwidth (in bytes) using the reported Upscaling Factor.
I/O BW Counter Correction Factor List Length	4	60	A value in this field defines I/O BW Counter Correction Factor List Length. Below are the valid values for the Correction Factor List Length: **0:** Do not apply a correction factor to the I/O BW Counter values. **1:** Apply a single correction factor specified in I/O BW Counter Correction Factor field to all the I/O BW Counter values (uniformly apply this correction factor to all data values retrieved from counters for all RMIDs). **Max RMID + 1:** If the value in this field matches the maximum supported RMID + 1 for this domain (as RMIDs are zero-indexed), indicated in RMDD:"Max RMID", apply the indicated indexed correction factor specified in MBM Correction Factor list to the corresponding the RMID value for the I/O BW counter.
I/O BW Counter Correction Factor []	-	64	A list of I/O BW Counter Correction Factors. The list will contain zero, one or Max RMID + 1 entries. Fixed point 32-bit format per entry in this list. Counter values may be multiplied by the correction factor to account for processor-specific implementation variations.

5.4.11 Cache Allocation Registers for Device Agents Description Structure

A Cache Allocation Registers for Device Agents Description (CARD) structure describes near cache allocation registers for Device Agents in a RDT domain.

There must be at least one instance of this structure for each RDT domain which supports I/O Cache Allocation Technology (I/O CAT). This structure is always contained within an RMDD structure.

Table 5-13. Cache Allocation Registers for Device Agents Description (CARD) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	10 - Cache Allocation Registers for Device Agents Description Structure
Length	2	2	Fixed: 40B
Reserved	4	4	Reserved(0)
Flags	4	8	• **Bit 0: Contention Bitmask Valid:** If Set, indicates 'Contention Bitmask' field is valid. Contention cache bitmask details are reported in 'Contention Bitmask' field. If Clear, indicates 'Contention Bitmask' field is not valid. • **Bit 1: Non-Contiguous Bitmasks Supported:** If Set, indicates non-contiguous capacity bitmasks are supported. The bits that are set in the various CAT Registers are not required to be contiguous. If Clear, non-contiguous bitmasks are not supported. • **Bit 2: Zero-length Bitmask:** If Set, indicates CAT Registers may be programmed with a value of zero, indicating zero Capacity Bitmask (CBM) bits set, and the associated CLOS will be prevented from allocating into the I/O L3 cache. If Clear, indicates CAT Registers do not support zero-length bitmasks, and at least one CBM bit must be set in the programmed mask. • Bits 3-31: Reserved.
Contention Bitmask	4	12	This field is valid if bit 0 (Contention Bitmask Valid) is set in flags field. Each set bit within the length of the bitmask (IO L3 Ways) indicates the corresponding unit (CBM bit) of the I/O L3 allocation may be used by other entities in the platform (e.g., an integrated graphics engine). Each unset bit within the length of the CBM indicates that the corresponding allocation unit can be used by an OS/VMM without interference from other integrated hardware agents in the system which may degrade determinism. Bits outside the length of the capacity bitmask are reserved.
Register Indexing Function Version	1	16	This field indicates Register Indexing Function Version Number. See Section 6.1.3.10 for details on the software usage guidance of this field.
Reserved	7	17	Reserved(0)
Register Base Address	8	24	Base address of Device Agent register set for this CARD. This address must be aligned according to the size of the register set size reported in

Field	Byte Length	Byte Offset	Description
			the Register Block Size field of this structure.
Register Block Size	4	32	Size of register space in units of number of 4KB pages. Registers are located in the range (X):(X+Y*4096), where X is the value reported in the Register Block Base Address field and Y is the value in this field. See Section 6.1.3.10 for details on the register layout.
CAT Register Offset for I/O	2	36	Bits 0-11: This field specifies the offset to the Cache Allocation registers for I/O in its corresponding 4KB page. If the register base address is X, and the value reported in this field is Y, the address for the CAT Registers for I/O is calculated as (X+Y). Each subsequent Cache Allocation register clump starts at the same Cache Allocation Register Offset in consecutive 4KB pages. Bits 12-15: Reserved(0)
CAT Register Block Size	2	38	Cache Allocation registers are a set of N adjacent 8-Byte sized registers, where N is the value specified in this field. The size of a Cache Allocation Register Block Size is thus 8*N bytes. Each Cache Allocation Register Block is organized in consecutive 4KB pages. Each Register Block for Cache Allocation starts at an offset specified by Cache Allocation Register Offset for I/O field in its corresponding 4KB page.

5.5 Memory Range and Region Mapping (MRRM) Structure Details

The top-level MRRM ACPI table is shown in the table below, and one instance of this table is defined at the system level, generated by the system BIOS. This table includes a unique signature and defines its variable length including all sub-structures.

The MRRM top-level structure describes host physical memory address ranges in the platform for region-ID mapping. The Region-Aware MBM and MBA features use these region IDs to enable monitoring and control per region-ID. Other features beyond RDT may use these same region numbers, that is, the region ID (e.g., “2”) used for a particular RDT feature maps identically to the region ID used for the other corresponding non-RDT feature, providing definitional symmetry. Specific memory ranges are defined and numbered via the Memory Range Entry (MRE) structure instances encoded within the MRRM structure.

As the MRRM table is fundamental to RDT Region Aware feature operation, if software encounters a Revision number that has not been enabled, then it

should cease to proceed forward and print an error message indicating that a software update is required.

Table 5-14. Memory Range and Region Mapping (MRRM) Structure

Field	Byte Length	Byte Offset	Description
Signature	4	0	"MRRM". Signature for the Memory Range and Region Mapping Structure
Length	4	4	Length, in bytes, of the description table including the length of the associated sub-structures.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID
OEM Table ID	8	16	For MRRM structure, the Table ID is the manufacturer model ID
OEM Revision	4	24	OEM Revision of MRRM Table for OEM Table ID.
Creator ID	4	28	Vendor ID of utility that created the table.
Creator Revision	4	32	Revision of utility that created the table.
Max Memory Regions Supported	1	36	Maximum number of **memory regions** that can be subject to Performance Monitoring, and Region-Aware Memory Bandwidth Monitoring and Allocation. One or more memory address ranges may be grouped to form memory regions.
Flags	1	37	**Bit 0: REGION_ASSIGNMENT_TYPE** ● If Clear, platform assigns a static region-ID for all memory ranges. When this bit is reported as clear, the Region-ID assigned for local accesses and remote accesses are provided in the Platform-assigned Local Region-ID field and Platform-assigned Remote Region-ID fields respectively of each Memory Range Entry. When this bit is reported as clear, the Region-ID programming registers field in each memory range entry must be 0. ● If Set, platform supports the capability for system software (OS/VMM) to assign region-IDs for local and remote accesses for each memory range. The registers for system software to program the region-IDs are enumerated in the Region-ID Programming Registers field of each Memory Range Entry. In this case, any initial platform-assigned Region-ID values may be read by software from the respective registers for each range. **Bits 1-7: Reserved(0).**
Reserved	26	38	Reserved (0).
Memory Range Entry List []	-	64	Array of one or more Memory Range Entries that each identify a contiguous host physical memory range to which memory bandwidth can be allocated and monitored. Refer to the Memory Range Entry structure definition.

5.5.1 Memory Range Entry (MRE) Structure

The Memory Range Entry (MRE) Structure hosts Memory Range Entries. Each Memory Range Entry identifies a contiguous host physical memory range to which memory bandwidth can be allocated and monitored. Each of these memory range entries provides the MMIO location of registers for software to configure Region-ID tagging for that memory range, if supported.

Table 5-15. Memory Range Entry (MRE) Structure

Field	Byte Length	Byte Offset	Description
Type	2	0	0 - Value of 0 in this field indicates this is a Memory Range Entry
Length	2	2	32B + sizeof (Region-ID Programming Registers[])
Reserved	4	4	Reserved(0)
Base Address Low	4	8	Low 32 Bits of the Base Address of the memory range
Base Address High	4	12	High 32 Bits of the Base Address of the memory range
Length Low	4	16	Low 32 Bits of the length of the memory range
Length High	4	20	High 32 Bits of the length of the memory range.
Region-ID Flags	2	24	Bit 0: Valid Local Region-ID • If Set, this host physical address memory range has valid Platform-assigned Static Local Region-ID. Bit 1: Valid Remote Region-ID • If Set, this host physical address memory range has valid Platform-assigned Static Remote Region-ID. Bits 2-15: Reserved.
Platform-assigned Static Local Region-ID	1	26	If REGION_ASSIGNMENT_TYPE bit in MRRM.Flags field is 0 and Valid Local Region-ID Flags is 1, this field enumerates the platform-assigned static region-ID for local accesses to this memory range.
Platform-assigned Static Remote Region-ID	1	27	If REGION_ASSIGNMENT_TYPE bit in MRRM.Flags field is 0 and Valid Remote Region-ID Flags is 1, this field enumerates the platform-assigned static region-ID for remote accesses to this memory range.
Reserved	4	28	Reserved (0).
Region-ID Programming Registers[]	-	32	If the REGION_ASSIGNMENT_TYPE bit in MRRM.Flags field is 1, this field specifies the registers to program Region-ID for this memory range. Host Physical Address of 8-Byte aligned RDT MMIO registers used to program the MBA/MBM Region-IDs of this range. Each Memory Range can be assigned two Region-IDs (a Local Region-ID for access by local socket agents and a Remote Region-ID for accesses by remote socket agents). One or more memory ranges can be grouped by into a region by assigning them the same Region-ID. Thus Region-IDs enable memory ranges to be organized into a set of regions that can be subject to

Field	Byte Length	Byte Offset	Description
			Memory Bandwidth Monitoring and Allocation. To support memory ranges that may be spanning multiple memory controllers, more than one register may be specified in this field. All registers identified in this field should be programmed identically. Refer to subsequent sections for further details and the architectural definition of these MBA/MBM Region-ID configuration registers.

Note that the base and length of each memory region may be used to cross-reference with memory regions defined in other ACPI tables such as HMAT and SRAT in a consistent fashion.

5.6 Architectural Intel® RDT Features for Non-CPU Agents (IRDT)

This section describes ACPI enumeration for architectural Intel RDT features for non-CPU agents. These features are predominantly enumerated via an ACPI structure for I/O RDT features with signature "IRDT". Note that while the existence of the IRDT object is sufficient to verify the presence of the I/O RDT feature on a processor, the revision of the IRDT table may change over time as the I/O interface and I/O bridge properties change. The encoded revision numbers can be used to manage this change over time.

5.6.1 RMID/CLOS tagging - ACPI Enumeration

5.6.1.1 ACPI Definitional Goals

A number of goals are accomplished through the IRDT ACPI enumeration definition in this chapter, including:

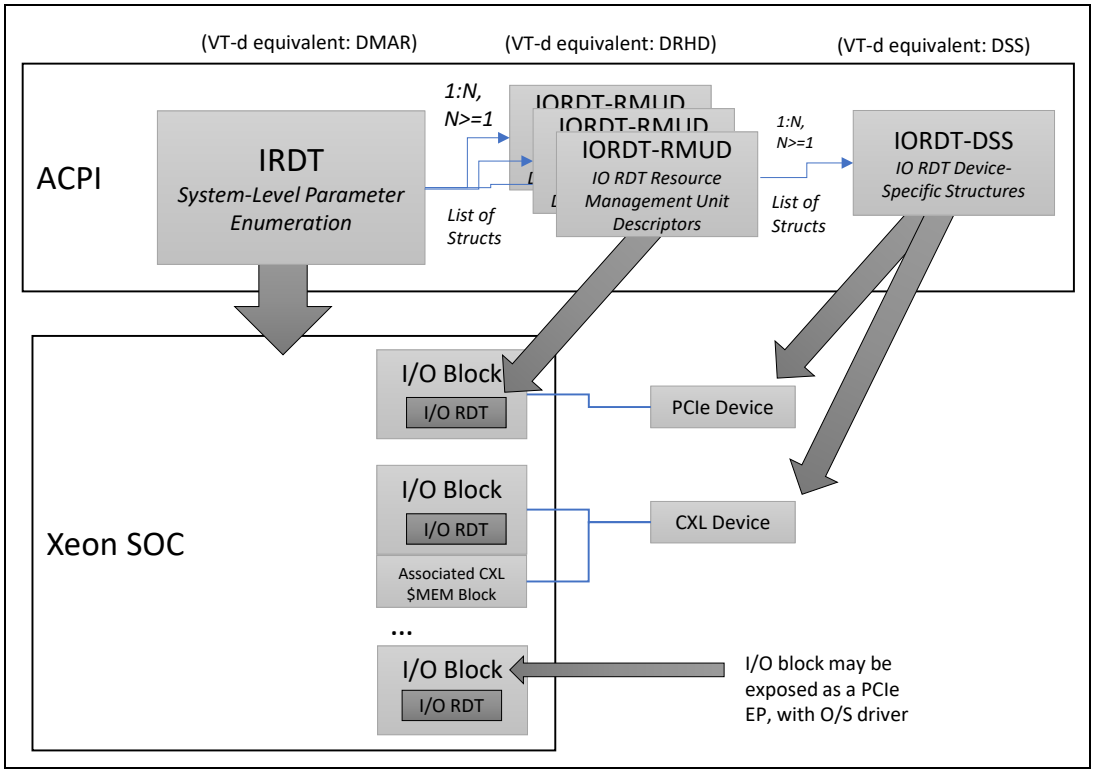
1. Providing top-level configuration information for the SoC, such as how many RMID/CLOS tags non-CPU agent Intel RDT supports relative to CPU agent Intel RDT (as enumerated by CPUID, see Chapter 3).
2. Providing a logical description of the control hierarchy – meaning which MMIO address to use to configure a link's RMID/CLOS tagging.
3. Provide flexibility in the implementation topology of devices behind I/O blocks, and cover cases with discrete or integrated PCIe and CXL links, and integrated accelerators.
4. Provide enhanced ease-of-use information for software, including device topologies, TC/VC/Channel mapping information for advanced QoS usages for forward-compatibility.

5.6.1.2 IRDT ACPI Enumeration Overview

This section provides a number of diagrams introducing key I/O Intel RDT structures and their mapping to Intel SoC components. Section 5.6.1.4 provides table specifics.

The top-level ACPI structure defined to support I/O Intel RDT is the "IRDT" structure. This is a vendor-specific extension to the ACPI table space [4]. The named IRDT structure is generated by BIOS and contains all other non-CPU agent Intel RDT ACPI enumeration structures and fields as described in this chapter.

Figure 5-3. Non-CPU Agent Intel® RDT ACPI Enumeration



Note that all Reserved fields in IRDT structures should be initialized to 0 by BIOS.

Under the IRDT structure in the hierarchy (embedded within the IRDT structure) are the I/O Intel RDT Resource Management Unit Descriptors (RMUDs.). The RMUDs typically map to I/O blocks within the system, though it is possible that one RMUD may be defined at other levels (such as one RMUD per SoC).

An example mapping is shown in Figure 5-3, showing ACPI details at the top, and Intel® Xeon® SoC mappings to hardware blocks at the bottom. The IRDT and RMUD relationships are shown for a typical implementation, in which RMUDs describe the properties of an I/O block. The IRDT table defines zero or more RMUDs, and an RMUD contains one of more RPs.

The RMUD structures contain two embedded structures, the Device Specific Structures (DSSes) and Resource Control Structures (RCSes) which map to devices and links and help describe the relationships regarding which I/O devices are connected to particular links, and which I/O links are in use by which devices. Each RMUD defines one or more DSS and RCS structures.

In the example of Figure 5-3, one DSS exists per PCIe, CXL or other non-CPU agent device (including accelerators), subservient to an RMUD. A CXL device may be expected to have multiple links (for example, CXL.Cache and CXL.IO) and this topology is described by the associated DSS structure and multiple RCS structures for the device and its links. Note that Figure 5-3 shows the DSS structure downstream of the RMUD but does not show the RCS for simplicity.

Figure 5-4 shows an example of the RMUD mapping to DSS and RCS structures. Each device attached to an I/O block is described by a DSS, and has one or more links, with properties described in the RCS structures. The RCS structures contain pointers to MMIO locations (in absolute address form, not BAR-relative) to allow software to configure the RMID/CLOS tags and bandwidth shaping properties, if supported, in an I/O Block.

Figure 5-4. ACPI Enumeration – Detail of DSS and RCS Structures Downstream from an RMUD

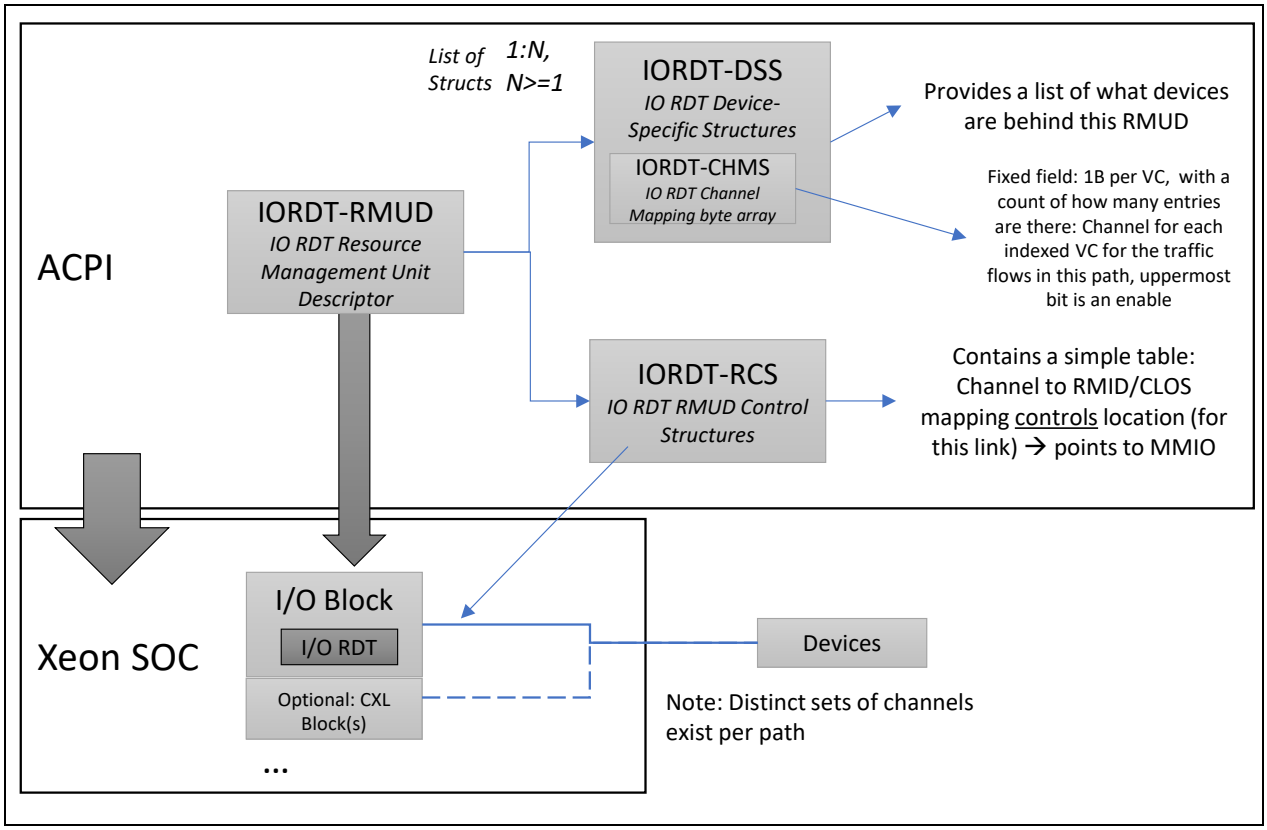
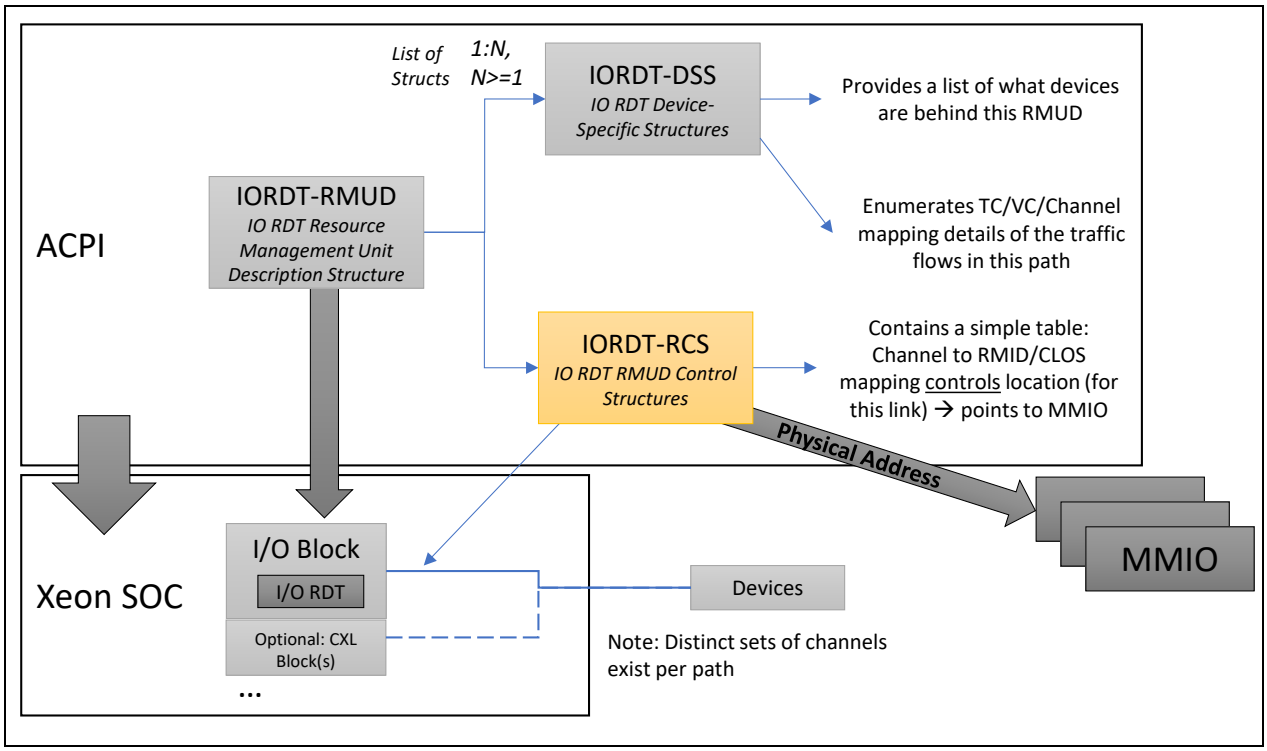


Figure 5-5 shows a further layer of detail where devices mapped through I/O blocks are described by the RMUDs, the DSS describes the properties of the device, and the RCS provides a pointer to the MMIO locations used for configuring the tagging and bandwidth shaping for a particular link.

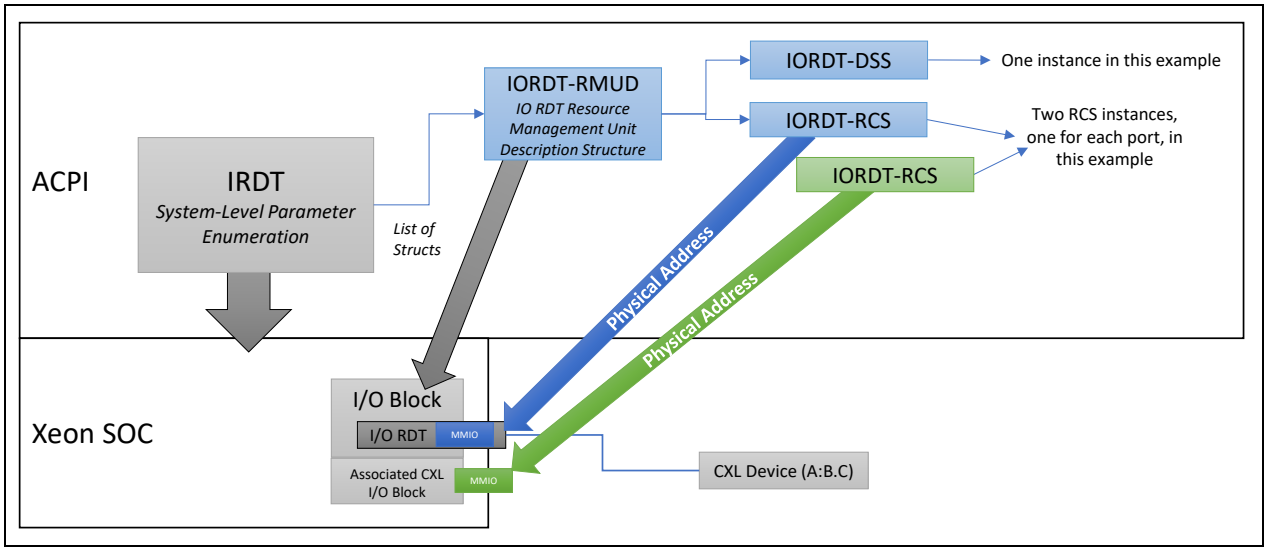
Figure 5-5. Mapping from RCS Structures to MMIO Addresses for Per-link Control



5.6.1.3 Example ACPI Enumeration Cases

Given the table hierarchy described in the preceding section, an example CXL Type 1 (CXL.IO + CXL.Cache) device mapping is shown in Figure 5-6. The device is described by one DSS behind an RMUD, while two RCSes are used, one for each link type (CXL.IO and CXL.Cache).

Figure 5-6. CXL Enumeration Example with CXL.IO and CXL.Cache Links



5.6.1.4 ACPI Feature Enumeration – Table Structure Details

5.6.1.4.1 Introduction and Notation

Given the previously described relationships of RMUD, DSS and RCS structures, table format details are described in this section.

Using the ACPI table hierarchy shown earlier in this chapter, following are the details of each table type and constituent fields. Field definitions are detailed in the table, and the text covers interpretation, corner cases, and interactions between fields.

5.6.1.4.2 IRDT Table Format and Field Descriptions

The top-level ACPI table, the I/O Resource Director Technology table (IRDT) is shown in Table 5-16, and one instance of this table is defined at the system level, generated by the system BIOS. This table includes a unique signature, and length including all sub-structures, including embedded RMUDs. The length of the IRDT table is variable.

Table 5-16. IRDT Table Format (Variable Length)

Field	Byte Length	Byte Offset	Description
Signature	4	0	"IRDT". Signature for the top-level I/O Intel RDT Description Table.
Length	4	4	Length, in bytes, of the description table including the length of the associated remapping structures.
Revision	1	8	1
Checksum	1	9	Checksum: Entire table must sum to zero.
OEMID	6	10	OEM ID.
OEM Table ID	8	16	For IORDT description table, the Table ID is the manufacturer model ID.
OEM Revision	4	24	OEM Revision of IRDT Table for OEM Table ID.
Creator ID	4	28	Vendor ID of utility that created the table.
Creator revision	4	32	Revision of utility that created the table.
IO Protocol Flags	2	36	Bit 0: IO_PROTO_MON -- Set if I/O Intel RDT Monitoring capabilities are supported somewhere on the platform for I/O protocol devices. Bit 1: IO_PROTO_CTL -- Set if I/O Intel RDT Allocation capabilities are supported somewhere on the platform for I/O protocol devices. Bit 2-15 : Reserved.

Field	Byte Length	Byte Offset	Description
Cache Protocol Flags	2	38	Bit 0: IO_COH_MON -- Set if I/O Intel RDT Monitoring capabilities are supported somewhere on the platform for coherent non-IA agents. Bit 1: IO_COH_CTL -- Set if I/O Intel RDT Allocation capabilities are supported somewhere on the platform for coherent non-CPU agents. Bit 2-15 : Reserved.
Reserved	8	40	-
Resource Management Hardware Blocks[]	-	48	A list of structures. The list will contain one or more Resource Management Unit Descriptors (RMUDs). The RMUD structure is described next.

A series of high-level flags allows the basic capabilities of monitoring and control for I/O links (for example, PCIe) and coherent links (for example, CXL) to be quickly extracted. Embedded within the IRDT table is a set of one or more Resource Management Unit Descriptor Structures (RMUDs), which are typically mapped to I/O blocks and define their properties. In some instantiations, one RMUD may be defined for the system, or in a finer-grained approach, one RMUDs may be defined for each downstream link and device combination, though this is expected to be an uncommon case.

5.6.1.4.3 RMUD Table Format and Field Descriptions

The Resource Management Unit Descriptor (RMUD) structure, definition is shown in Table 5-17, and includes a number of fields including length of the RMUD instance and all embedded sub-structures (DSS and RCS entries), an integration parameter that map to the SoC properties, including the minimum and maximum RMID and CLOS tags that are available for use in monitoring and controlling devices under this RMUD. While the common case is that these parameters would match the CPU agent Intel RDT parameters, there may be certain RMUDs which support a subset of the overall RMID and CLOS space.

Table 5-17. RMUD Table Format (Variable length)

Field	Byte Length	Byte Offset	Description
Type	1	0	Type 0 = “RMUD”. Signature for the I/O Intel RDT Resource Management Unit Descriptor.
Reserved	3	1	Reserved.
Length	4	4	Total length of this RMUD and all sub-structures.
Segment	2	8	The PCI Segment containing this RMUD, and all of the devices that are within it.
Reserved	3	10	Reserved.

Field	Byte Length	Byte Offset	Description
DSS and RCS Structures []	---	13	List of devices behind this RMUD, with one DSS table instance per device. Contains a list of DSS control structures and RCS control structures, identified by their "Type" field at offset zero in the sub-structures. The DSS and RCS structures described next.

Each RMUD entry contains a number of embedded DSS and RCS structures, identified by their "Type" fields, which describe the devices and links behind a given RMUD.

5.6.1.4.4 DSS Table Format and Field Descriptions

The Device Scope Structures behind each RMUD describe the properties of a device, that is, each DSS maps 1:1 with a device behind a particular RMUD. The DSS table definition is shown in Table 5-18, including a "type" field (Type = 0 identifies a DSS), the length of the entry, device type, and an embedded channel management structure (CHMS). The CHMS defines which RCS(es) are applicable to controlling this device (DSS), and which internal I/O block Channels each of the link's virtual channels (VCs) may map to (in the case of PCIe, up to eight VCs are supported, but only the first entry is valid in the case of CXL). Valid configurations for the CHMS include one entry per RCS (link).

In the DSS Device Type field, a value of 0x02 denotes that a PCIe Sub-hierarchy is described by this DSS. Each root port described by a DSS will have type 0x02. System software may use the enumerated devices found under such a root port to comprehend share bandwidth relationships in the channels under an RMUDS.

DSS type 0x01 indicates the presence of a root complex integrated endpoint device (RCEIP), such as an accelerator. Note that a PCI sub-hierarchy may denote a root port, and for every DSS that corresponds to a root port it is expected that Device Type = 0x2.

Note that the CHMS field contains a list of CHMS structures, which may describe for instances DSS entries which are capable of sending traffic over multiple channels (which are in turn described by unique RCS entries).

Note that no discrete pluggable devices (for example, PCIe cards) are directly described by the DSS entries, rather the root ports are indicated (Device Type 0x2).

Fields described in this DSS table are only valid when the Revision value is 1 in the top-level IRDT structure. Refer to section 5.6.1.4.5 (DSS Table format) and following for cases when Revision value is 2 or above in the top-level IRDT structure.

Table 5-18. DSS Table Format (Variable length)

Field	Byte Length	Byte Offset	Description
Type	2	0	0 = DSS
Length	2	2	Length of this Entry in Bytes.
Device Type	1	4	The following values are defined for this field. 0x01: Root Complex Integrated Endpoint (RCEIP) Device - The device identified by the 'Path' field is a root complex integrated PCI endpoint device. 0x02: PCI Sub-hierarchy - The device identified by the 'Path' field is a PCI-PCI bridge. In this case, the specified bridge device and all its downstream devices are included in the scope. Other values for this field are reserved for future use.
Enumeration ID	2	5	If Device Type equals 1 or 2, this field lists the BDF
Reserved	1	7	Reserved
Structure: CHMS and RCS Enumeration []	---	8	Packed as byte fields. One RCS may support multiple DSSes, and one DSS may have multiple RCSs (links), so this is an array, with size derivable from the DSS Length field. Within each entry: **Byte 0:** RCS Enumeration ID controlling this link. Corresponds to the enumeration ID of the RCS structure under this DSS. **Bytes 1-8:** Represents the index into the "RCS-CFG-Table" used by the corresponding VC. Byte 1 represents the channel for VC0, Byte 2 represents the channel for VC1, and so on. In this field, bit 7 is a valid bit (entry is not valid if enable bit is cleared). Bit 6, when set, indicates that this channel is shared with another DSS. The number of valid bytes in this field is defined in the per-RCS "Channel Count" field, any unused bytes (for example, for a single-Channel CXL link) are Reserved. **See text below for version-specific interpretation.** **Bytes 9-15:** Reserved (padding)

5.6.1.4.5 DSS Table Format for IRDT Table Revision 2

When revision 2 is specified in the IRDT table, the Channel Count field in the DSS structure indicates how many links the RCS has been bifurcated into and the lowest number channel can be used to control lowest number BDF and so on. This controls register indexing pattern.

5.6.1.4.6 RCS Table Format and Field Descriptions

The RCS structure provides details of the type of monitoring and controls supported for a particular link interface type, such as PCIe or CXL, and an MMIO location in which a table exists that can be used to apply monitoring and control features. The MMIO location provided is absolute location in MMIO space (64 bits), rather than hosted in a particular device and defined relative to a BAR.

Table 5-19. RCS Table Format (v1, Currently 40B)

Field	Byte Length	Byte Offset	Description
Type	2	0	RCS = 1.
Length	2	2	Length, in bytes, of the description table including the length of the associated remapping structures.
Link Interface Type	2	4	Type of link interface: 0x0 = PCIe or CXL.IO 0x1 = CXL.Cache 0x2 and above: Reserved
RCS Enumeration ID	1	6	A unique identifier for this RCS under this RMUD.
Channel Count	1	7	Number of Channels defined for this link interface (affects the interpretation of the CHMS structure within the corresponding DSS).
Flags	2	8	Bit 0: Reserved. Bit 1: RTS: RMID Tagging supported. Bit 2: CTS: CLOS Tagging Supported. Bit 3: REGW: if set, the RMID and CLOS defined in the RCS Block MMIO locations are 2B registers. If clear, they are 8B registers. Bits 4-15: Reserved.
RMID Block Offset	2	10	Byte offset from the RCS Block MMIO Location where the RMID tagging fields begin.
CLOS Block Offset	2	12	Byte offset from the RCS Block MMIO Location where the CLOS tagging fields begin.
Reserved	18	14	Reserved.
RCS Block MMIO Location	8	32	RCS Hosting I/O Block MMIO BAR Location defines an MMIO physical address.

Fields mentioned in this RCS table are only valid when the Revision value is 1 in top-level IRDT structure. Refer section 5.6.1.4.7 RCS Table format when Revision value 2 or above in top-level IRDT structure.

Note that if CXL.IO and PCIe devices share the bandwidth of a certain RCS and its channels, then traffic for both protocols is carried on the same channel entries.

Note that in the enumeration the fields, the RMID offset, and CLOS offset are specified relative to the "RCS Block MMIO Location" field, meaning that the RMID and CLOS offsets may be relocatable within the MMIO space. The offset defines the block of a contiguous set of RMID or CLOS tagging fields, and the number of entries is defined by the "Channel Count" field (for example, a value of 8 channels may be common in certain PCIe tagging implementations). Note that if CXL.IO and PCIe devices share the bandwidth of a certain RCS and its channels, then traffic for both protocols is carried on the same channel entrie

Note that in the enumeration the fields, the RMID offset, and CLOS offset are specified relative to the "RCS Block MMIO Location" field, meaning that the RMID and CLOS offsets may be relocatable within the MMIO space. The offset defines the block of a contiguous set of RMID or CLOS tagging fields, and the number of entries is defined by the "Channel Count" field (for example, a value of 8 channels may be common in certain PCIe tagging implementations).

5.6.1.4.7 RCS Table Format for Revision 2

These fields are only valid when the Revision value is 2 in top-level IRDT structure.

Table 5-20. RCS Table Format (v2, Currently 40B)

Field	Byte Length	Byte Offset	Description
Type	2	0	RCS = 1.
Length	2	2	Length, in bytes, of the description table including the length of the associated remapping structures.
Link Interface Type	2	4	Type of link interface: 0x0 = PCIe or CXL.IO 0x1 = CXL.Cache 0x2 and above: Reserved
RCS Enumeration ID	1	6	A unique identifier for this RCS under this RMUD.
Channel Count	1	7	Number of Channels defined for this link interface (affects the interpretation of the CHMS structure within the corresponding DSS).
Flags	2	8	Bit 0: Reserved. Bit 1: RTS: RMID Tagging supported. Bit 2: CTS: CLOS Tagging Supported. Bit 3: REGW: if set, the RMID and CLOS defined in the RCS Block MMIO locations are 2B registers. If clear, they are 4B registers. Bit 4: CXLD: if set, indicates that more than one CXL device resides behind the I/O link represented by this RCS, for instance due to link bifurcation. This has implications on the interpretation of the Channel Count field. See the surrounding text for details.

Field	Byte Length	Byte Offset	Description
			Bits 5-15: Reserved.
RMID Block Offset	2	10	Byte offset from the RCS Block MMIO Location where the RMID tagging fields begin.
CLOS Block Offset	2	12	Byte offset from the RCS Block MMIO Location where the CLOS tagging fields begin.
Reserved	18	14	Reserved.
RCS Block MMIO Location	8	32	RCS Hosting I/O Block MMIO BAR Location defines an MMIO physical address.

Channel Count indicates how many links the RCS has been bifurcated into and the lowest number channel can be used to control lowest number BDF and so on. This controls register indexing pattern. When set, RCS::Flags::CXLD (Bit 4) is a special case, where Channel Count field means something specific for a CXL bifurcated device in that if software detects more than one BDF within the scope of this DSS and it is enumerated PCIe Bridge, then there will be multiple devices under the scope of single RCS. In that case, these devices will be implicitly sharing bandwidth in an some way, such as sharing a bifurcated CXL physical interface. This bandwidth sharing may also apply to PCIe physical devices or functions within a single PCIe physical device but is not represented by the CXLD bit.

5.7 Model-Specific Intel® RDT Features for CPU Agents

This section describes BIOS configuration options for Model-Specific Intel RDT features for CPU agents.

5.7.1 BIOS Configuration for Resource Aware MBA

See Appendix A.3 for Resource Aware MBA processor support details. See Appendix B.1.1 for Resource Aware MBA feature details. Note that Resource Aware MBA is a distinct feature from *Region Aware MBA*.

The Resource-aware MBA feature is a model-specific extension to the Third Generation of MBA (Chapter 3) which provides a set of extended capabilities to better handle heterogenous memory types on complex modern SoCs. A model-specific implementation is used as memory types may change significantly over

the course of time. A more detailed description of Resource Aware MBA is provided in the next chapter.

To support Resource Aware MBA, the system BIOS shall support a legacy BW profile configuration knob with a drop-down menu of three options as with Second-Generation MBA.

- MBA BW profile
  - Linear(default)
  - Biased
  - Legacy

In addition, BIOS shall add three knobs with a drop-down menu for Resource-Aware MBA in particular. These scaling ratios enable tuning of MBA calibration values to the typical bandwidth levels available from each type of heterogeneous downstream memory type, and tuning values may be further scaled by the number of memory channels or links populated with each type of memory. An example implementation of this tuning code will be provided with the Intel Reference BIOS implementation for each applicable platform.

1. Description: "PMM BW downscaling vs the baseline Total memory BW profile. For example: picking 1/2x at results in scaling PMM BW throttling in a 2:1 ratio versus DDR throttling."
   - PMM MBA BW downscale
     - 1x (default)
     - 1/2x
     - 1/4x
     - 1/8x
2. Description: "CXL (Type3) BW downscaling vs the baseline Total memory BW profile. For example: picking 1/2x results in scaling CXL (Type3) BW throttling in a 2:1 ratio versus DDR throttling."
   - CXL (Type3) MBA BW downscale
     - 1x (default)
     - 1/2x
     - 1/4x
     - 1/8x
3. Description: "Remote Target BW downscaling vs the baseline Total memory BW profile. For example: picking 1/2x results in scaling Remote Target BW throttling in a 2:1 ratio versus DDR throttling."
   - Remote Target MBA (UPI) BW downscale
     - 1x (default)
     - 1/2x
     - 1/4x
     - 1/8x

§

6 MMIO Register Descriptions

This chapter describes the Intel RDT related MMIO registers. As described in previous chapters, traditional interfaces such as MSRs are discussed in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.

6.1 Enhanced Intel® RDT Register Location

Referencing the ACPI object definitions of Chapter 5, the register set (MMIO interfaces) for each Resource Monitoring Domain Description (RMDD) structure in the platform is mapped in a 4KB-aligned memory mapped page. The exact location of the register region for each feature is implementation dependent and is communicated to system software by BIOS through the ACPI ERDT and MRRM reporting structures (described in Chapter 5, *BIOS Considerations*).

6.1.1 Software Access to Registers

The following sections describe software access conventions to MMIO-based RDT registers, including register indexing functions, bitfield definitions and properties.

6.1.2 Register Attributes

The following table defines the attributes used in the RDT feature Registers. The registers are discussed in Section 6.1.3.

Table 6-1. Register Attributes Definitions

Attribute	Description
RW	Read-Write field that may be either set or cleared by software to the desired state.
RO	Read-only field that cannot be directly altered by software
RsvdP	"Reserved and Preserved" field that is reserved for future RW implementations. Registers are read-only and must return 0 when read. **Important:** Software must preserve the value read for subsequent writes during read-modify-write (RMW) operations.

6.1.3 Register Descriptions

The following table summarizes the RDT feature memory-mapped registers. The scope of these registers is per RMDD structure.

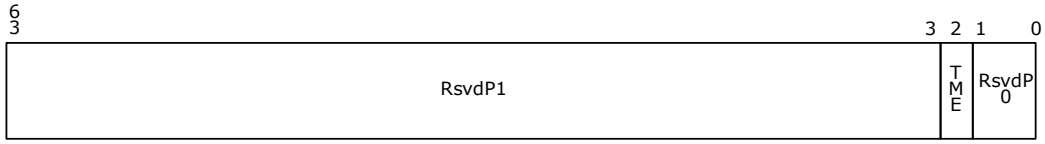
Table 6-2. Memory-Mapped Register Block Reference

	Register Name	Size(b)	Description
1	RDT CTRL	64	Register to control RDT MBM and MBA features.
2	Cache Monitoring Register for CPU Agents	64	Register reporting cache occupancy telemetry for CPU Agents. MMIO Base address of this register is specified in CMRC sub-structure of ERDT APCI. Field name: CMT Register Block Base Address for CPU. See later sections for data interpretation.
3	Memory-bandwidth Monitoring Registers for CPU Agents	64	Register reporting memory bandwidth monitoring telemetry data for CPU Agents. MMIO Base address of this register is specified in MMRC sub-structure of ERDT APCI. Field name: MBM Register Block Base Address. See later sections for data interpretation.
4	Optimum Memory-bandwidth Allocation Registers for CPU Agents	64	Register to configure optimum memory bandwidth allocation targets for CPU Agents. MMIO Base address of this register is specified in MARC sub-structure of ERDT APCI. Field name: MBA Optimal BW Register Block Base Address.
5	Minimum Memory-bandwidth Allocation Registers for CPU Agents	64	Register to configure minimum memory bandwidth allocation targets for CPU Agents. MMIO Base address of this register is specified in MARC sub-structure of ERDT APCI. Field name: MBA Minimum BW Register Block Base Address.
6	Maximum Memory-bandwidth Allocation Registers for CPU Agents	64	Register to configure maximum memory bandwidth allocation targets for CPU Agents. MMIO Base address of this register is specified in MARC sub-structure of ERDT APCI. Field name: MBA Maximum BW Register Block Base Address.
7	Cache Monitoring Registers for Non-CPU Agents	64	Register reporting cache occupancy telemetry for Non-CPU Agents. MMIO Base address of this register is specified in CMRD sub-structure of ERDT APCI. Field name: Register Base Address. See later sections for data interpretation.
8	Cache Allocation Registers for Non-CPU Agents	64	Register to configure cache allocation rules for CPU Agents. MMIO Base address of this register is specified in CARD sub-structure of ERDT APCI. Field name: Register Base Address.

	Register Name	Size(b)	Description
9	Total I/O Bandwidth Registers for Non-CPU Agents	64	Register reporting Total I/O bandwidth telemetry for Non-CPU Agents. MMIO Base address of this register is specified in IBRD sub-structure of ERDT APCI. Field name: Register Base Address. See later sections for data interpretation.
10	I/O Miss Bandwidth Registers for Non-CPU Agents	64	Register reporting I/O Miss bandwidth telemetry for Non-CPU Agents. MMIO Base address of this register is specified in IBRD sub-structure of ERDT APCI. Field name: Register Base Address. See later sections for data interpretation.
11	Region-ID Programming Registers[]		Register to configure range to region mapping via system software (OS/VMM). MMIO Base address of this register is specified in MRRM ACPI. Field name: Region-ID Programming Registers[].

6.1.3.1 RDT Control Register for CPU Agents

Figure 6-1. RDT Control Register



Abbreviation	RDT_CTRL
General Description	Register to configure RDT features for CPU Agents
Indexing Function	N/A
Address	RMDD.Control Register Base Address
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63:3	RO	0h	RsvdP1: Reserved and Preserved	Reserved
2:2	RW	1h	TME: Total Mode En	Total Mode Enable: 1: Indicates Total MBM and MBA Mode to enable the Legacy MSR interfaces.

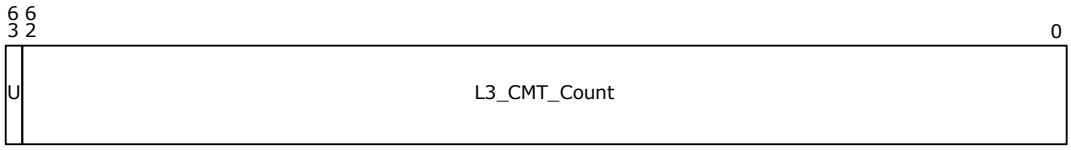
				0: Indicates Per Region Aware MBM and MBA to opt-in to using the MMIO Register interfaces for Region Aware RDT features.
1:0	RO	0h	RsvdP0: Reserved and Preserved	Reserved

Software should enable region aware MBA and MBM to prior to usages of the Region Aware MBA and MBM features, via the RDT CTRL register on a per-RMDD basis. These registers should be programmed identically across all RMDD instances (e.g., by programming each RDT_CTRL MMIO register indicated in each RMDD) for CPU agents. It is recommended that software use Region Aware MBM when Region Aware MBA is enabled and vice versa. Mixed mode use (e.g., legacy MSR interfaces for MBM with Region Aware MBA or vice versa) is not supported and may lead to inconsistent behavior.

For total bandwidth monitoring and allocation software may continue to use MSR interfaces by setting the TME bit (Total Mode En Bit[2]) to 1. MSR interfaces should not be used if Total Mode En Bit[2] is clear. Legacy MSR interfaces do not offer Region Aware Memory bandwidth monitoring and allocation.

6.1.3.2 Cache Monitoring Register for CPU Agents

Figure 6-2. CMT Register



Abbreviation	L3_CMT_RMID_n n: Refer to ACPI ERDT for MAX RMID. RMIDs are zero-referenced. Hence, this range will encompass 0 to ("MAX RMID" reported by RMDD sub-structure).
General Description	Register to report Cache Occupancy for CPU Agents
Indexing Function	See Section 6.1.3.2.1
RMID Address	CMRC.CMT Register Block Size for CPU + Indexing function mentioned above
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63	RO	0h	U: Unavailable	• 0: Indicates data for this RMID is available and monitored for the resource or RMID. • 1: Indicates data for this RMID is not available or not monitored for the resource or RMID, and bits(62:0) should be ignored.
62:0	RO	0h	L3_CMT_Count	The value in this field indicates Cache Monitoring (occupancy) telemetry. See later sections in Chapter 7, *Programming Guidelines*, for data interpretation.

6.1.3.2.1 RMID Organization in CMT Register Block

Software should use the RMID indexing algorithm discussed in this section only if the "Register Indexing Function Version" field value is 1 in the CMRC sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

RMIDs are organized in sequential fashion in the CMT Register Blocks. Software may consult the CMRC sub-structure from ERDT in ACPI when retrieving CMT telemetry using CMT Register Block Base Address for CPU, CMT Register Block Size for CPU, CMT Register Clump Size for CPU CMT Register Clump Stride for CPU fields of the CMRC sub-structure. Each block size is 4KB. CMT registers are located in the range `(CMT Register Block Base Address): (CMT Register Block Base Address + CMT Register Block Size Value x 4096)`. To index RMIDs in the block use the following pseudocode algorithm, where "%" represents the modulo operator and "/*** … ***/" is used to encapsulate comments):

```
MMIO_ADDRESS_for_RMID# = CMT Register Block Base Address +
((RMID# / CMT Register Clump Size for CPU) x CMT Register Clump
Stride for CPU) + ((RMID# % CMT Register Clump Size for CPU) x
8B);


/*** MMIO_ADDRESS_for_RMID# < (CMT Register Block Base Address +
CMT Register Block Size Value x 4096) ***/
```

Here,

Input Parameter: RMID#

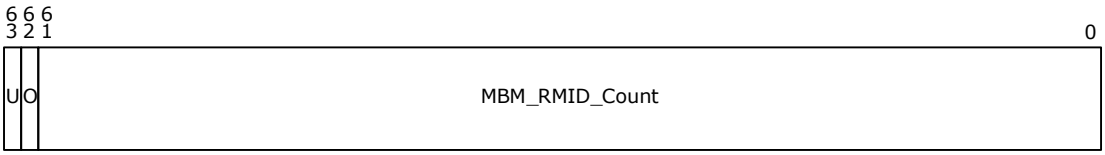
Parameters for Indexing:

- "CMT Register Block Base Address" field reported by CMRC sub-structure of ERDT ACPI.

- "CMT Register Block Size Value" reported by CMRC sub-structure of ERDT ACPI.
- Max RMIDs supported on the platform reported by RMDD sub-structure of ERDT ACPI.
- "CMT Register Clump Size for CPU" and "CMT Register Clump Stride for CPU" fields values to be enumerated by CMRC sub-structure.

6.1.3.3 Memory Bandwidth Monitoring Registers for CPU Agents

Figure 6-3. Per Region Per RMID MBM Register



Abbreviation	MBM_Region_m_RMID_n Variable "m": Refer to ACPI MRRM to find out number of regions supported. This range will be 0 to ("Max Memory Regions Supported" reported by MRRM ACPI -1) Variable "n": Refer ACPI ERDT for MAX RMID. RMIDs are zero-referenced. Hence, this range will be 0 to ("MAX RMID" reported by RMDD sub-structure).
General Description	Register to report Memory Bandwidth Monitoring for CPU Agents.
Indexing Function	See Section 6.1.3.3.1
RMID Address	MMRC.MBM Register Block Base Address + Indexing function mentioned above
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63	RO	0h	U: Unavailable	• 0: Indicates data for this RMID is available and monitored for the resource or RMID. 1: Indicates data for this RMID is not available or not monitored for the resource or RMID, and bits(61:0) should be ignored.

Bits	Access	Default	Field	Description
62	RO	0h	O: Overflow	• 0: Indicates that there is no overflow of the MBM counters • 1: Indicates that there is overflow of the MBM counters. It will be reset upon read, enabling a variable software-defined counter polling interval for reduced sampling overhead.
61:0	RO	0h	MBM_RMID_Count	The value in this field indicates Memory Bandwidth Monitoring telemetry. See later sections in Chapter 7, *Programming Guidelines*, for data interpretation.

6.1.3.3.1 RMID Organization in MBM Register Block

Software should use the RMID indexing algorithm discussed in this section only if "Register Indexing Function Version" field value is 1 in MMRC sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

RMIDs are organized in interleaved fashion in the MBM Register Blocks. Software may consult the MMRC sub-structure from ERDT ACPI for retrieving MBM registers using MBM Register Block Base Address and MBM Register Block Size. Each block size is 4KB. MBM registers are located in the range (`MBM Register Block Base Address): (MBM Register Block Base Address + MBM Register Block Size Value x 4096B)`. To index RMIDs in the block per Region use the following pseudocode algorithm:

```
Block_to_locate_RMID# = ((RMID# % 32) / 8) x 4 x 4096B;

Offset_within_this_Block = ((((RMID#/32)x8)+RMID#%8) x
8B)+(Region# x 2048B);


MMIO_ADDRESS_for_RMID#_Region# =

MBM Register Block Base Address + Block_to_locate_RMID# +
Offset_within_this_Block;


/*** MMIO_ADDRESS_for_RMID#_Region# < (MBM Register Block Base
Address + MBM Register Block Size Value *4096B) ***/
```

Here,

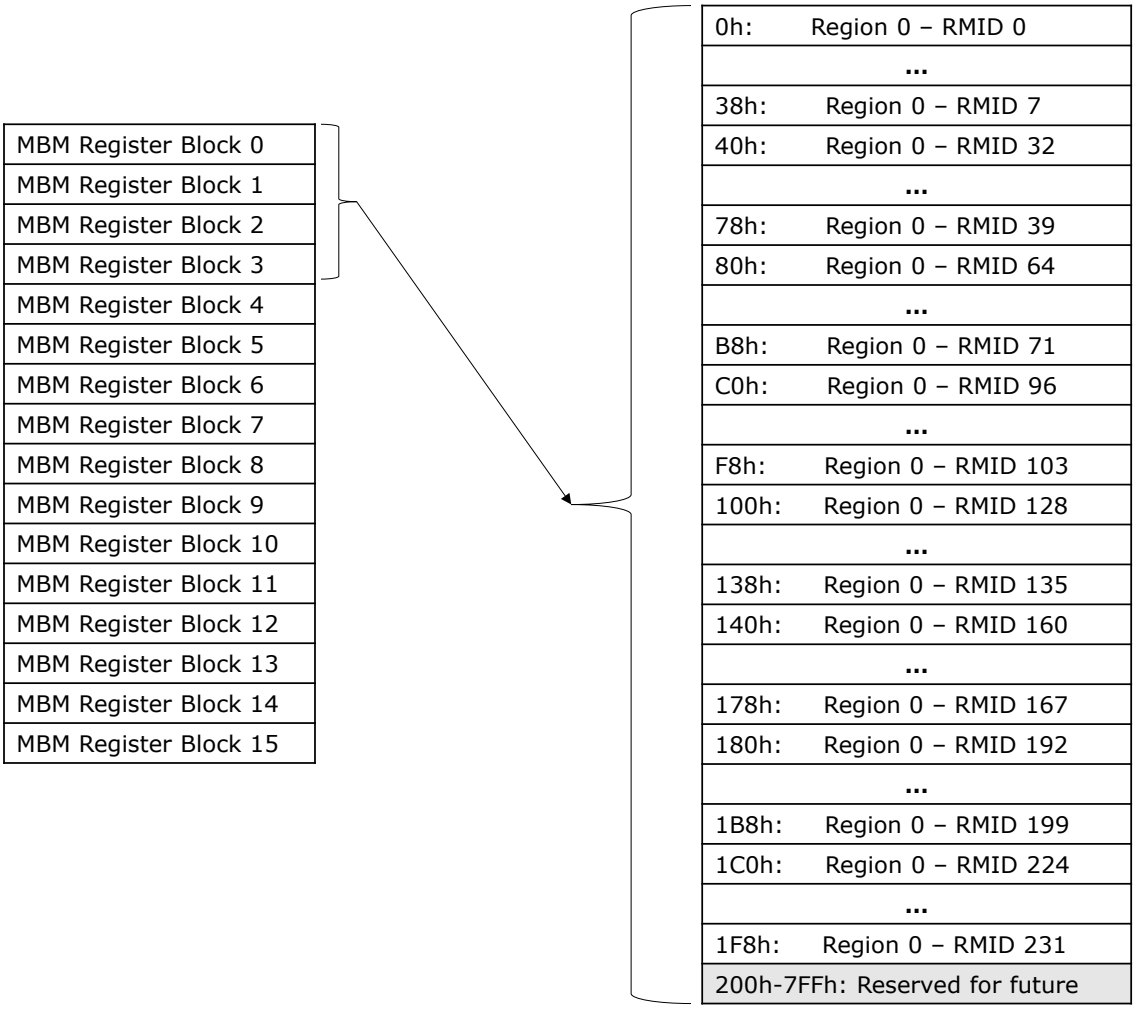
Input Parameter: RMID# and Region#

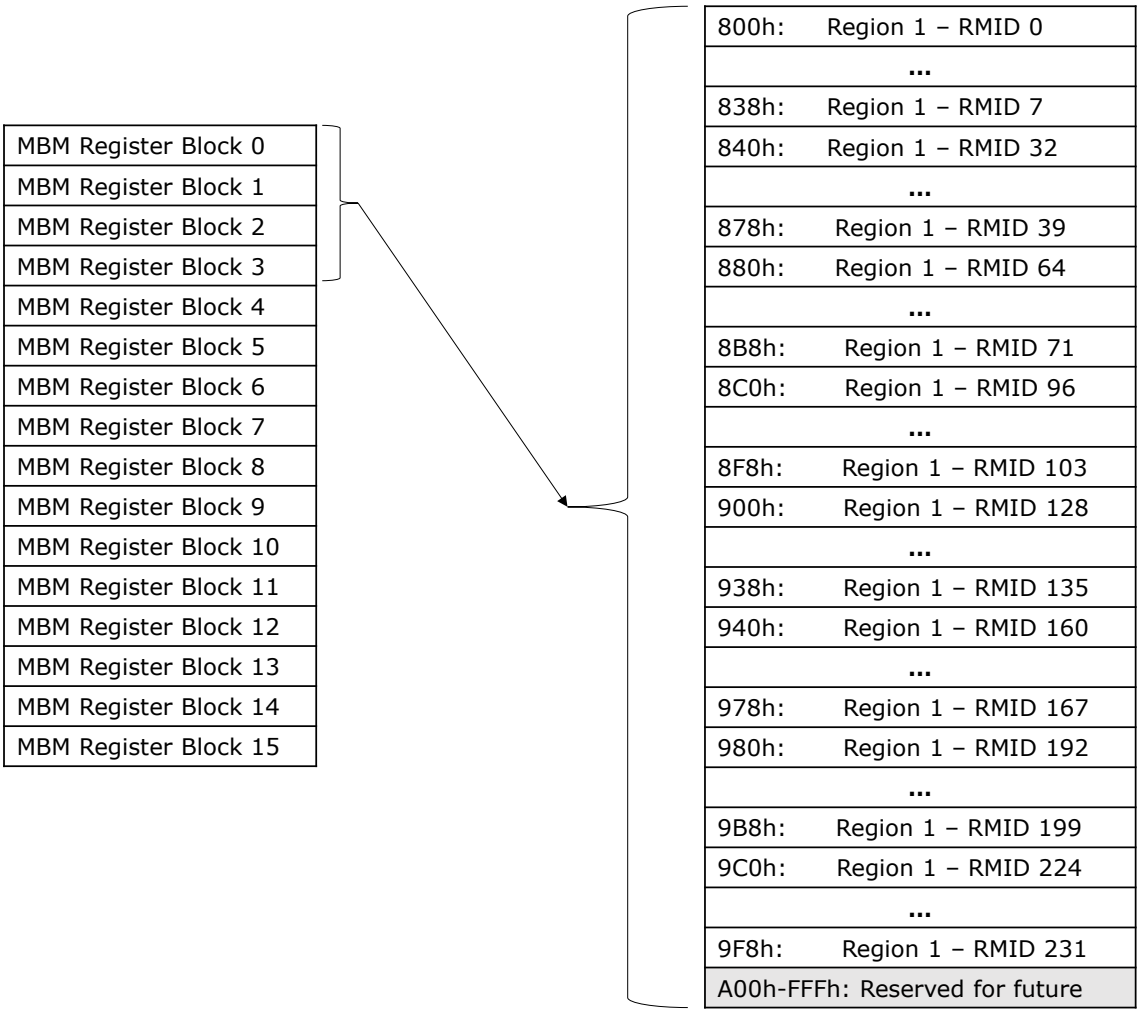
Parameters for Indexing:

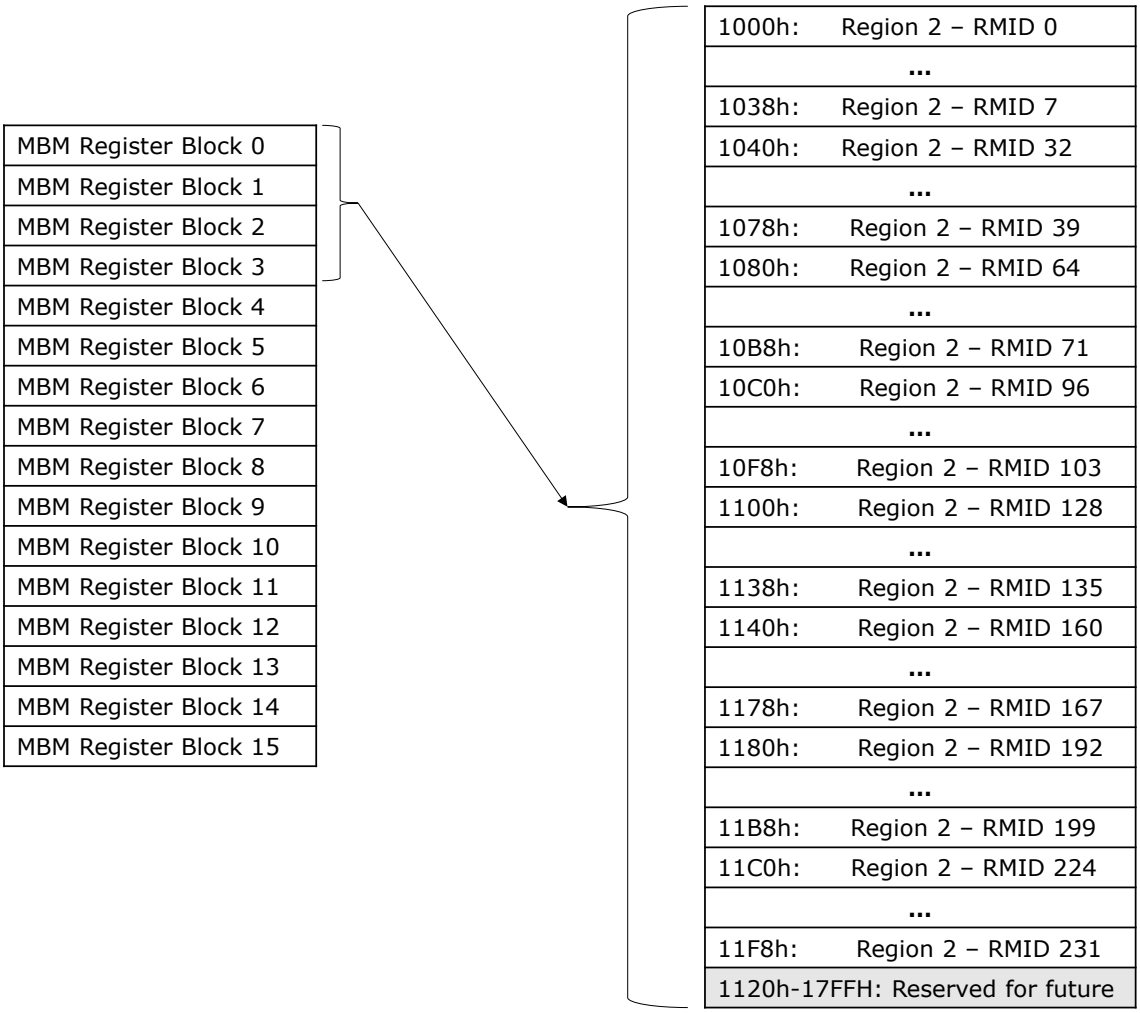
- "MBM Register Block Base Address" field reported by MMRC sub-structure of ERDT ACPI.
- "MBM Register Block Size Value" reported by MMRC sub-structure of ERDT ACPI.
- Max RMIDs supported on the platform reported by RMDD sub-structure of ERDT ACPI.
- Max Regions support on the platform reported by MRRM ACPI.

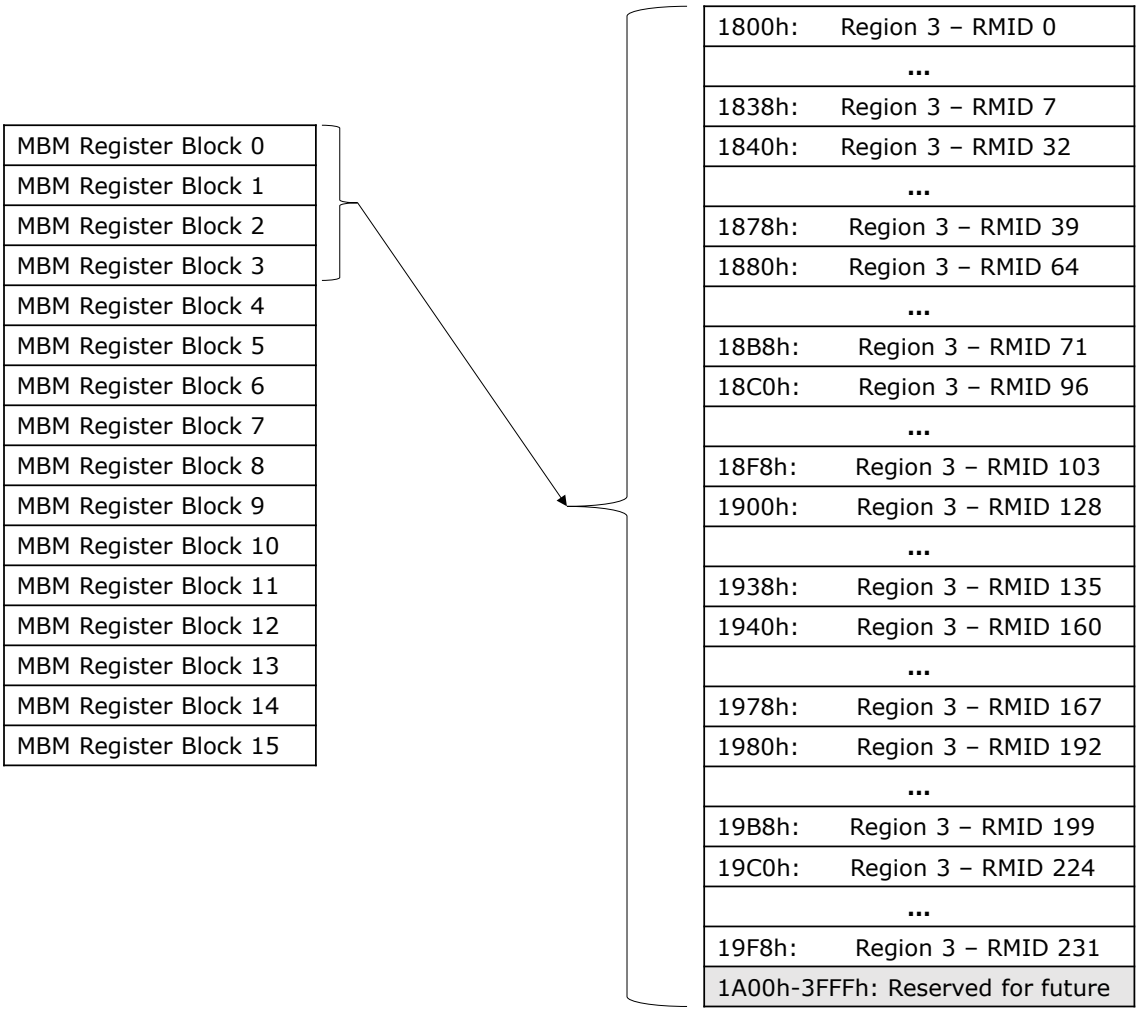
An example of MBM register blocks is described below in Figure 6-4.

Figure 6-4. Interleaved RMID MBM Register



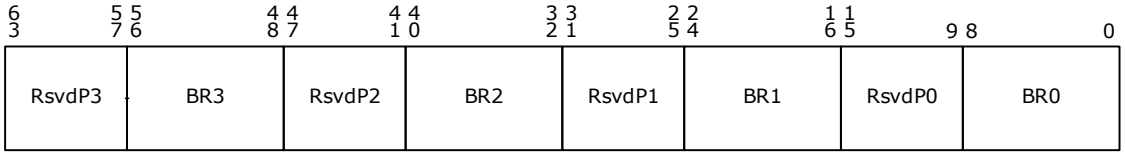

MBM Register Block 0
MBM Register Block 1
MBM Register Block 2
MBM Register Block 3
MBM Register Block 4
MBM Register Block 5
MBM Register Block 6
MBM Register Block 7
MBM Register Block 8
MBM Register Block 9
MBM Register Block 10
MBM Register Block 11
MBM Register Block 12
MBM Register Block 13
MBM Register Block 14
MBM Register Block 15
800h: Region 1 – RMID 0
...
838h: Region 1 – RMID 7
840h: Region 1 – RMID 32
...
878h: Region 1 – RMID 39
880h: Region 1 – RMID 64
...
8B8h: Region 1 – RMID 71
8C0h: Region 1 – RMID 96
...
8F8h: Region 1 – RMID 103
900h: Region 1 – RMID 128
...
938h: Region 1 – RMID 135
940h: Region 1 – RMID 160
...
978h: Region 1 – RMID 167
980h: Region 1 – RMID 192
...
9B8h: Region 1 – RMID 199
9C0h: Region 1 – RMID 224
...
9F8h: Region 1 – RMID 231
A00h-FFFh: Reserved for future


MBM Register Block 0
MBM Register Block 1
MBM Register Block 2
MBM Register Block 3
MBM Register Block 4
MBM Register Block 5
MBM Register Block 6
MBM Register Block 7
MBM Register Block 8
MBM Register Block 9
MBM Register Block 10
MBM Register Block 11
MBM Register Block 12
MBM Register Block 13
MBM Register Block 14
MBM Register Block 15
1000h: Region 2 – RMID 0
...
1038h: Region 2 – RMID 7
1040h: Region 2 – RMID 32
...
1078h: Region 2 – RMID 39
1080h: Region 2 – RMID 64
...
10B8h: Region 2 – RMID 71
10C0h: Region 2 – RMID 96
...
10F8h: Region 2 – RMID 103
1100h: Region 2 – RMID 128
...
1138h: Region 2 – RMID 135
1140h: Region 2 – RMID 160
...
1178h: Region 2 – RMID 167
1180h: Region 2 – RMID 192
...
11B8h: Region 2 – RMID 199
11C0h: Region 2 – RMID 224
...
11F8h: Region 2 – RMID 231
1120h-17FFH: Reserved for future



6.1.3.4 Optimal Memory Bandwidth Allocation Register for CPU Agents

Figure 6-5. MBA Optimal Bandwidth Register



Abbreviation	MBA_OPTIMAL_BW_n n: Refer to ACPI ERDT for Max CLOS. CLOS are zero-referenced. Hence, this range will be 0 to ("MAX CLOS" reported by the ERDT top-level structure).

General Description	Register to configure Optimal Bandwidth Control Window for Memory Bandwidth Allocation per CLOS.
Indexing Function	See Section 6.1.3.4.1
CLOS Address	MARC.MBA Register Block Base Address + Indexing function mentioned above
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63:57	RsvdP	0h	RsvdP3: Reserved and Preserved	Reserved.
56:48	RW	1FFh	BR3: Bandwidth_Target_ Region 3	Optimal Bandwidth Control Value for Region 3. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
47:41	RsvdP	0h	RsvdP2: Reserved and Preserved	Reserved.
40:32	RW	1FFh	BR2: Bandwidth_Target_Region 2	Optimal Bandwidth Control Value for Region 2. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
31:25	RsvdP	0h	RsvdP1: Reserved and Preserved	Reserved.
24:16	RW	1FFh	BR1: Bandwidth_Target_ Region 1	Optimal Bandwidth Control Value for Region 1. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
15:9	RsvdP	0h	RsvdP0: Reserved and Preserved	Reserved.
8:0	RW	1FFh	BR0: Bandwidth_Target_Region 0	Optimal Bandwidth Control Value for Region 0. Ranges from 001h to 1FFh, with 001h as the minimum BW 1FFh as the maximum BW.

6.1.3.4.1 CLOS Organization in Optimal MBA Register Block

Software should use the CLOS indexing algorithm discussed in this section only if "Register Indexing Function Version" field value is 1 in MARC sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

Note that Region Aware MBA uses the same definition of "optimal", "min" and "max" bandwidth as described in earlier discussion in Section 3.2.4.4.3, *Optimal Bandwidth Caps per CLOS.*

CLOSs are organized in sequential fashion in the Optimal MBA Register Blocks. Software may consult the MARC sub-structure from ERDT ACPI for configuring the per-agent per-region per-CLOS optimum target bandwidth using the MBA Optimal BW Register Block Base Address and MBA Register Block Size fields of MARC sub-structure. Each block size is 4KB. Optimum MBA registers are located in the range `(MBA Optimal BW Register Block Base Address): (MBA Optimal BW Register Block Base Address + MBA Register Block Size x 4096.` To index CLOSs per region in the block use the following pseudocode algorithm:

```
MMIO_ADDRESS_for_CLOS# = MBA Optimal BW Register Block Base
Address + (Region# / 4) x 512B + CLOS# x 8B.

/*** MMIO_ADDRESS_for_CLOS# < (MBA Optimal BW Register Block Base
Address + MBA Register Block Size x 4096) ***/
```

Here,

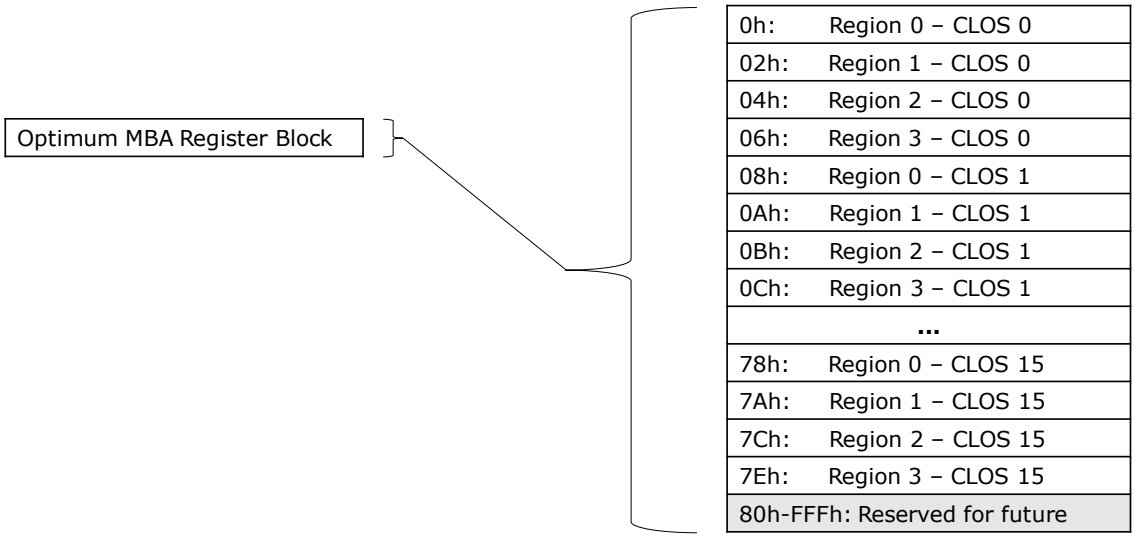
Input Parameter: CLOS#, Region# (multiple banks for registers 1st bank is for Region 0 to 3 and consecutively Region 4 to 7 after every 512B). This formula supports up to 64 CLOSs and an arbitrary number of regions.

Parameters for indexing:

- "MBA Optimal BW Register Block Base Address" field reported by MARC sub-structure of ERDT ACPI
- "MBA Register Block Size" reported by MARC sub-structure of ERDT ACPI
- Max CLOSs supported on the platform reported by ERDT ACPI.
- Max Regions support on the platform reported by MRRM ACPI.

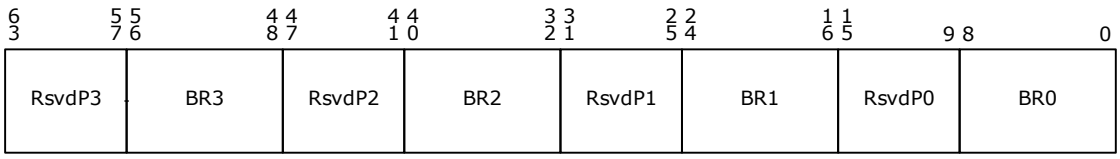
An example of Optimum MBA register blocks is described below in Figure 6-6.

Figure 6-6. Sequential CLOS arrangement in MBA Register



6.1.3.5 Minimum Memory Bandwidth Allocation Register for CPU Agents

Figure 6-7. Minimum MBA Register



Abbreviation	MBA_MINIMUM_BW_n n: Refer ACPI ERDT for Max CLOS. CLOSs are zero-referenced. Hence, this range will be 0 to ("MAX CLOS" reported by ERDT top-level structure).
General Description	Register to configure Minimum Bandwidth Control Window for Memory Bandwidth Allocation per CLOS.
Indexing Function	See Section 6.1.3.5.1
CLOS Address	MBA Minimum BW Register Block Base Address + Indexing function mentioned above
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63:57	RsvdP	0h	RsvdP3: Reserved and Preserved	Reserved.
56:48	RW	1FFh	BR3: Bandwidth for Region 3	Minimum Bandwidth Control Value for Region 3. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
47:41	RsvdP	0h	RsvdP2: Reserved and Preserved	Reserved.
40:32	RW	1FFh	BR2: Bandwidth for Region 2	Minimum Bandwidth Control Value for Region 2. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
31:25	RsvdP	0h	RsvdP1: Reserved and Preserved	Reserved.
24:16	RW	1FFh	BR1: Bandwidth for Region 1	Minimum Bandwidth Control Value for Region 1. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
15:9	RsvdP	0h	RsvdP0: Reserved and Preserved	Reserved.
8:0	RW	1FFh	BR0: Bandwidth for Region 0	Minimum Bandwidth Control Value for Region 0. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.

6.1.3.5.1 CLOS Organization in Minimum MBA Register Block

Software should use CLOS indexing algorithm discussed in this section only if "Register Indexing Function Version" field value is 1 in MARC sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

Note that Region Aware MBA uses the same definition of "optimal", "min" and "max" bandwidth as described in earlier discussion in Section 3.2.4.4.3, *Optimal Bandwidth Caps per CLOS.*

CLOSs are organized in sequential fashion in the Minimum MBA Register Blocks. Software may consult the MARC sub-structure from ERDT ACPI for configuring the per-agent per-region per-CLOS minimum target bandwidth using the MBA Minimum BW Register Block Base Address and MBA Register Block Size fields of the MARC sub-structure. Each block size is 4KB. Minimum MBA registers are located in the range (MBA Minimum BW Register Block Base Address): (MBA Minimum BW Register Block Base Address + MBA Register Block Size x 4096). To index CLOSs per region in the block use the following pseudocode algorithm:

MMIO_ADDRESS_for_CLOS# = MBA Minimum BW Register Block Base Address + (Region# / 4) x 512B + CLOS# x 8B.

/*** MMIO_ADDRESS_for_CLOS# < (MBA Minimum BW Register Block Base Address + MBA Register Block Size x 4096) ***/

Here,

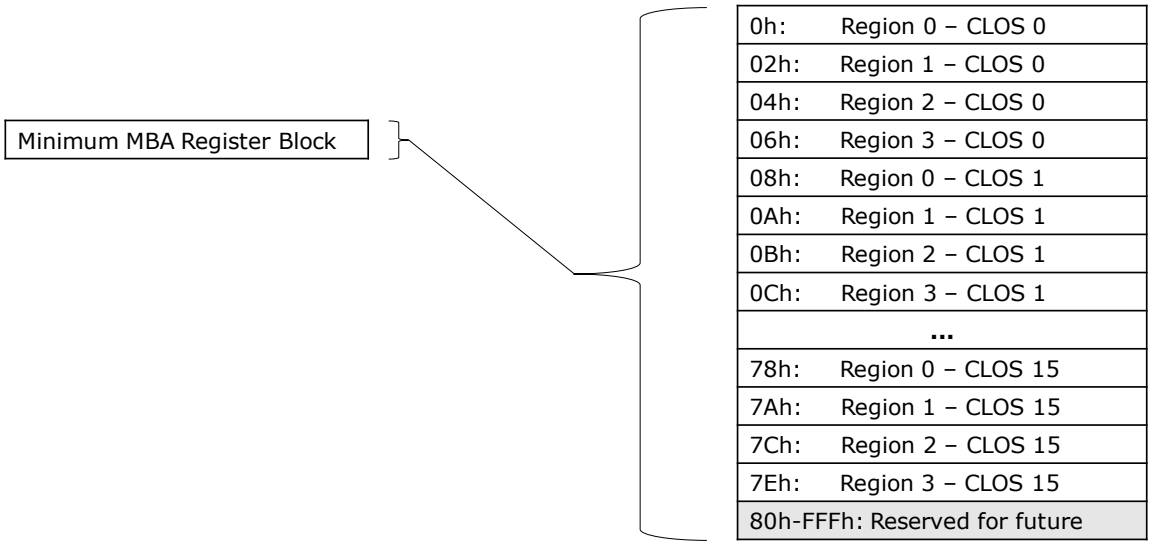
Input Parameter: CLOS#, Region# (multiple banks for registers 1st bank is for Region 0 to 3 and consecutively Region 3 to 7 after every 512B). This formula supports up to 64 CLOSs and an arbitrary number of regions.

Parameters for Indexing:

- "MBA Minimum BW Register Block Base Address" field reported by MARC sub-structure of ERDT ACPI
- "MBA Register Block Size" reported by MARC sub-structure of ERDT ACPI
- Max CLOSs supported on the platform reported by ERDT ACPI.
- Max Regions support on the platform reported by MRRM ACPI.

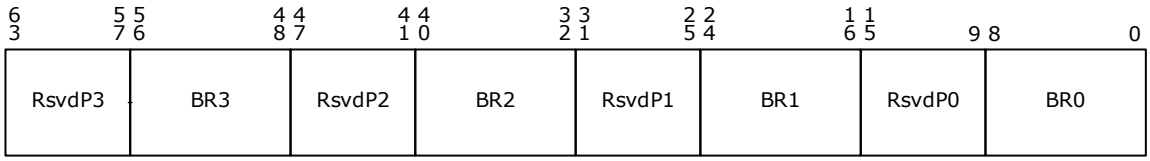
An example of Minimum MBA register blocks is described below in Figure 6-8.

Figure 6-8. Sequential CLOS arrangement in MBA Register



6.1.3.6 Maximum Memory Bandwidth Allocation Registers for CPU Agents

Figure 6-9. Maximum MBA Register



Abbreviation	MBA_MAXIMUM_BW_n n: Refer to ACPI ERDT for Max CLOS. CLOSs are zero-referenced. Hence, this range will be 0 to ("MAX CLOS" reported by ERDT top-level structure).
General Description	Register to configure Maximum Bandwidth Control Window for Memory Bandwidth Allocation per CLOS.
Indexing Function	See Section 6.1.3.6.1
CLOS Address	MBA Maximum BW Register Block Base Address + Indexing function mentioned above.
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63:57	RsvdP3	0h	RsvdP3: Reserved and Preserved	Reserved.
56:48	RW	1FFh	BR3: Bandwidth for Region 3	Maximum Bandwidth Control Value for Region 3. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
47:41	RsvdP2	0h	RsvdP2: Reserved and Preserved	Reserved.
40:32	RW	1FFh	BR2: Bandwidth for Region 2	Maximum Bandwidth Control Value for Region 2. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
31:25	RsvdP1	0h	RsvdP1: Reserved and Preserved	Reserved.
24:16	RW	1FFh	BR1: Bandwidth for Region 1	Maximum Bandwidth Control Value for Region 1. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.
15:9	RsvdP0	0h	RsvdP0: Reserved and Preserved	Reserved.
8:0	RW	1FFh	BR0: Bandwidth for Region 0	Maximum Bandwidth Control Value for Region 0. Ranges from 001h to 1FFh, with 001h as the minimum BW and 1FFh as the maximum BW.

6.1.3.6.1 CLOS Organization in Maximum MBA Register Block

Software should use CLOS indexing algorithm discussed in this section only if "Register Indexing Function Version" field value is 1 in MARC sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

Note that Region Aware MBA uses the same definition of "optimal", "min" and "max" bandwidth as described in earlier discussion in Section 3.2.4.4.3, *Optimal Bandwidth Caps per CLOS.*

CLOSs are organized in sequential fashion in the Maximum MBA Register Blocks. Software may consult the MARC sub-structure from the ERDT ACPI for configuring per-agent per-region per-CLOS maximum target bandwidth using the MBA Maximum BW Register Block Base Address and MBA Register Block Size fields of the MARC sub-structure. Each block size is 4KB. Maximum MBA registers are located in the range (MBA Maximum BW Register Block Base Address): (MBA Maximum BW Register Block Base Address + MBA Register Block Size x 4096). To index CLOS per region in the block use the following pseudocode algorithm:

```
MMIO_ADDRESS_for_CLOS# = MBA Maximum BW Register Block Base Address + (Region# / 4) x 512B + CLOS# x 8B.

/*** MMIO_ADDRESS_for_CLOS# < (MBA Maximum BW Register Block Base Address + MBA Register Block Size x 4096) ***/
```

Here,

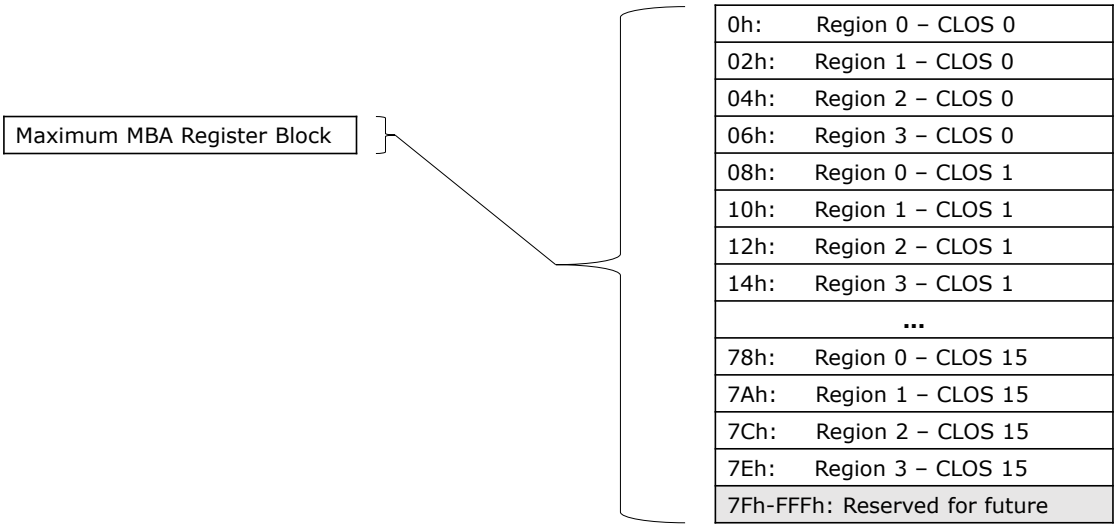
Input Parameter: CLOS#, Region# (multiple banks for registers 1st bank is for Region 0 to 3 and consecutively Region 3 to 7 after every 512B). This formula supports up to 64 CLOSs and an arbitrary number of regions.

Parameters for Indexing:

- "MBA Maximum BW Register Block Base Address" field reported by MARC sub-structure of ERDT ACPI
- "MBA Register Block Size" reported by MARC sub-structure of ERDT ACPI
- Max CLOSs supported on the platform reported by ERDT ACPI.
- Max Regions support on the platform reported by MRRM ACPI.

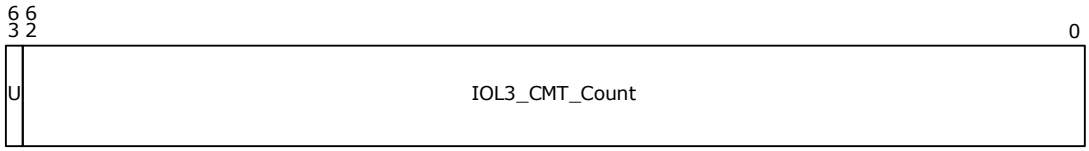
An example of Maximum MBA register blocks is described below in Figure 6-10.

Figure 6-10. Sequential CLOS arrangement in MBA Register



6.1.3.7 Cache Monitoring Registers for Non-CPU Agents

Figure 6-11. CMT Register



Abbreviation	IOL3_CMT_RMID_n n: Refer ACPI ERDT for MAX RMID. RMIDs are zero-referenced. Hence, this range will be 0 to ("MAX RMID" reported by RMDD sub-structure).
General Description	Register to report Cache Occupancy for Non-CPU Agents
Indexing Function	See Section 6.1.3.7.1
RMID Address	CMRD.Register Base Address + Indexing function mentioned above.
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63	RO	0h	U: Unavailable	• 0: Indicates data for this RMID is available and monitored for the resource or RMID. • 1: Indicates data for this RMID is not available or not monitored for the resource or RMID, and bits(62:0) should be ignored.
62:0	RO	0h	IOL3_CMT_Count	The value in this field indicates Cache Monitoring (occupancy) telemetry for Non-CPU agents. See later sections in Chapter 7, *Programming Guidelines*, for data interpretation.

6.1.3.7.1 RMID Organization in CMT Register Blocks

Software should use RMID indexing algorithm discussed in this section only if the "Register Indexing Function Version" field value is 1 in the CMRD sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

RMIDs are organized in sequential fashion in the CMT Register Blocks. Software may consult the CMRD sub-structure from ERDT ACPI for retrieving CMT telemetry using the the Register Base Address, Register Block Size, CMT Register Offset for I/O and CMT Register Clump Size for I/O fields of the CMRD sub-structure. Each block size is 4KB. CMT registers are located in the range (X):(X+Y x 4096), where X is value reported in Register Base Address field and Y is the value reported in Register Block Size field. To index RMIDs in the block the following pseudocode algorithm may be used:

```
MMIO_ADDRESS_for_RMID# = Register Base Address + ((RMID# / CMT
Register Clump Size for I/O) x 4096B) + CMT Register Offset for
I/O + ((RMID# % CMT Register Clump Size for I/O) x 8B);

/*** MMIO_ADDRESS_for_RMID# < (Register Base Address + Register
Block Size x 4096) ***/
```

Here,

Input Parameter: RMID#

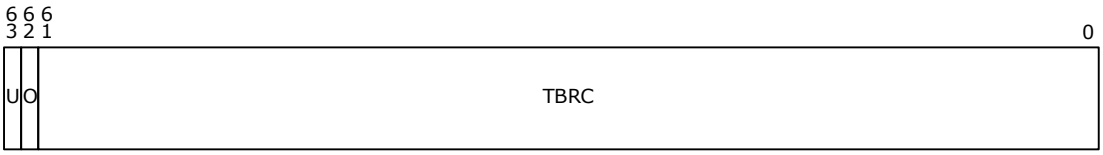
Parameters for Indexing:

- "Register Base Address" field reported by CMRD sub-structure of ERDT ACPI
- "Register Block Size" reported by CMRD sub-structure of ERDT ACPI
- Max RMID supported on the platform reported by RMDD sub-structure of ERDT ACPI.

- "CMT Register Offset for I/O" and "CMT Register Clump Size for I/O" fields value reported by CMRD sub-structure of ERDT ACPI

6.1.3.8 Total I/O Bandwidth Monitoring Registers for Non-CPU Agents

Figure 6-12. Total I/O Bandwidth Register



Abbreviation	Total_IO_BW_RMID_n n: Refer ACPI ERDT for MAX RMID. RMIDs are zero-referenced. Hence, this range will be 0 to ("MAX RMID" reported by RMDD sub-structure).
General Description	Register to report Per RMID Total IO Bandwidth to the near cache.
Indexing Function	See Section 6.1.3.8.1
RMID Address	IBRD.Register Base Address + Indexing function mentioned above.
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63	RO	0h	U: Unavailable	• 0: Indicates data for this RMID is available and monitored for the resource or RMID. • 1: Indicates data for this RMID is not available or not monitored for the resource or RMID, and bits(61:0) should be ignored.
62	RO	0h	O:Overflow	• 0: Indicates that there is no overflow of the Total IO BW counters. • 1: Indicates that there is overflow of the Total IO BW counters. It will be reset upon read, enabling a variable software-defined counter polling interval for reduced sampling overhead.

Bits	Access	Default	Field	Description
61:0	RO	0h	TBRC: Total_IO_BW_RMID_Count	The value in this field indicates Total IO Bandwidth telemetry. See later sections in Chapter 7, *Programming Guidelines*, for data interpretation.

6.1.3.8.1 RMID Organization in Total I/O BW Register Blocks

Software should use the RMID indexing algorithm discussed in this section only if the "Register Indexing Function Version" field value is 1 in IBRD sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

RMIDs are organized in sequential fashion in the Total I/O BW Register Blocks. Software may consult the IBRD sub-structure from ERDT ACPI for retrieving Total I/O BW telemetry using the Register Base Address, Register Block Size, Total I/O BW Register Offset and Total I/O BW Register Clump Size fields of the IBRD sub-structure. Each block size is 4KB. Total I/O BW registers are located in the range (X):(X+Y*4096), where X is value reported in the Register Base Address field and Y is the value reported in Register Block Size field. To index RMIDs in the block the following pseudocode algorithm may be used:

```
MMIO_ADDRESS_for_RMID# = Register Base Address + ((RMID#/ "Total I/O BW Register Clump Size") x 4096B) + "Total I/O BW Register Offset" + ((RMID# % "Total I/O BW Register Clump Size") x 8B);

/*** MMIO_ADDRESS_for_RMID# < (Register Base Address + Register Block Size x 4096) ***/
```

Here,

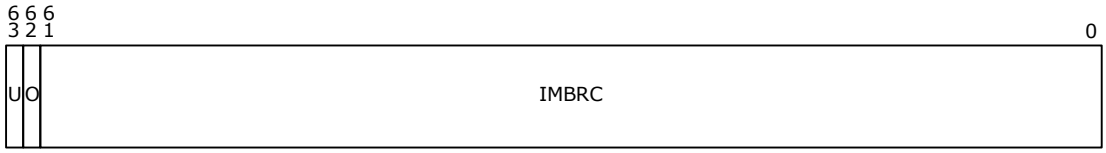
Input Parameter: RMID#

Parameters for Indexing:

- "Register Base Address" field reported by IBRD sub-structure of ERDT ACPI
- "Register Block Size" reported by IBRD sub-structure of ERDT ACPI
- Max RMID supported on the platform reported by RMDD sub-structure of ERDT ACPI.
- Total I/O BW Register Offset and Total I/O BW Register Clump Size fields value reported by IBRD sub-structure of ERDT ACPI

6.1.3.9 I/O Miss Bandwidth Monitoring Registers for Non-CPU Agents

Figure 6-13. I/O Miss Bandwidth Register



Abbreviation	IO_MISS_BW_RMID_n n: Refer ACPI ERDT for MAX RMID. RMIDs are zero-refenced. Hence, this range will be 0 to ("MAX RMID" reported by RMDD sub-structure).
General Description	Register to report Per RMID IO Bandwidth monitoring for Misses from the near cache.
Indexing Function	See Section 6.1.3.9.1
RMID Address	IBRD.Register Base Address + Indexing function mentioned above.
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63	RO	0h	U: Unavailable	• 0: Indicates data for this RMID is available and monitored for the resource or RMID. 1: Indicates data for this RMID is not available or not monitored for the resource or RMID, and bits(61:0) should be ignored.
62	RO	0h	O:Overflow	• 0: Indicates that there is no overflow of the IO BW Miss counters. • 1: Indicates that there is overflow of the IO BW Miss counters. It will be reset upon read, enabling a variable software-defined counter polling interval for reduced sampling overhead.
61:0	RO	0h	IMBRC: IO_MISS_BW_RMID_Count	The value in this field indicates I/O Miss Bandwidth telemetry. See later sections in Chapter 7, ***Programming Guidelines***, for data interpretation.

6.1.3.9.1 RMID Organization in I/O Miss BW Register Blocks

Software should use the RMID indexing algorithm discussed in this section only if the "Register Indexing Function Version" field value is 1 in IBRD sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

RMIDs are organized in sequential fashion in the I/O Miss BW Register Blocks. Software may consult the IBRD sub-structure from ERDT ACPI for retrieving Total I/O BW telemetry using the Register Base Address, Register Block Size, I/O Miss BW Register Offset and I/O Miss Register Clump Size fields of the IBRD sub-structure. Each block size is 4KB. I/O Miss BW registers are located in the range (X):(X+Y*4096), where X is value reported in Register Base Address field and Y is the value reported in Register Block Size field. To index RMIDs in the block the following pseudocode algorithm may be used:

```
MMIO_ADDRESS_for_RMID# = Register Base Address + ((RMID#/ "I/O Miss Register Clump Size") x 4096B) + "I/O Miss BW Register Offset" + ((RMID# % "I/O Miss Register Clump Size") x 8B);

/*** MMIO_ADDRESS_for_RMID# < (Register Base Address + Register Block Size x 4096) ***
```

Here,

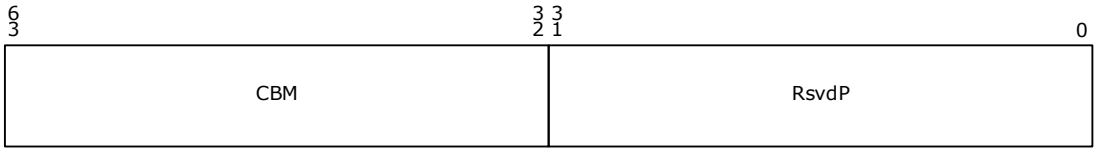
Input Parameter: RMID#

Parameters for Indexing:

- "Register Base Address" field reported by IBRD sub-structure of ERDT ACPI
- "Register Block Size" reported by IBRD sub-structure of ERDT ACPI
- Max RMIDs supported on the platform reported by RMDD sub-structure of ERDT ACPI.
- I/O Miss BW Register Offset and I/O Miss Register Clump Size fields value reported by IBRD sub-structure of ERDT ACPI

6.1.3.10 Cache Allocation Registers for Non-CPU Agents

Figure 6-14. CAT_IO_REG Register



Abbreviation	IOL3_MASK_n n: Refer to ACPI ERDT for Max CLOS. CLOSs are zero-referenced. Hence, this range will be 0 to ("MAX CLOS" reported by ERDT top-level structure).
General Description	Register to configure I/O L3 cache way mask per CLOS.
Indexing Function	See Section 6.1.3.10.1
CLOS Address	CARD.Register Base Address + Indexing Function mentioned above.
Scope	Per Resource Management Domain (Per RMDD)

Bits	Access	Default	Field	Description
63:32	RW	xxh	CBM: Capacity Bit Mask	Software may use this field to update cache capacity bitmask per CLOS. Bitmask length can be determined via Field "Number of IO L3 ways" in the ERDT ACPI table.

Bits	Access	Default	Field	Description
31:0	RsvdP	0h	RsvdP: Reserved and Preserved: Reserved	Reserved.

6.1.3.10.1 CLOS Organization in CAT Register Blocks

Software should use the CLOS indexing algorithm discussed in this section only if "Register Indexing Function Version" field value is 1 in the MARC sub-structure. Software should be upgraded to handle any versions > 1 in this field which would be defined in future version of this specification.

CLOS are organized in sequential fashion in the register blocks. Software may consult the CARD sub-structure from ERDT ACPI for retrieving CAT configuration details includign Register Base Address, Register Block Size, CAT Register Offset for I/O and CAT Register Block Size fields of CARD sub-structure. Each block size is 4KB. CAT registers are located in the range (X):(X+Y x 4096), where X is the value reported in the Register Base Address field and Y is the value reported in Register Block Size field. To index CLOS in the block the following pseudocode algorithm may be used:

Note: All the MMIO registers identified for the CLOS# should be programmed identically in each block.

```
For ( i = 0 to Register Block Size)

{

 if (CLOS# <= CAT Register Block Size)

 {

       ARRAY_OF_MMIO_ADDRESS_for_CLOS#[] = Register Base Address + "CAT
 Register Offset for I/O" + (CLOS# x 8B) + (4096 x i)

 }

}
```

Here,

Input Parameter: CLOS#

Parameters for Indexing:

- "Register Base Address" field reported by CARD sub-structure of ERDT ACPI
- "Register Block Size" reported by CARD sub-structure of ERDT ACPI
- Max CLOSs supported on the platform reported by ERDT ACPI.

- CAT Register Offset for I/O and CAT Register Block Size fields value reported by CARD sub-structure of ERDT ACPI

6.1.3.11 Region-ID Programming Registers[]

Not defined. The Region ID Programming Registers field in MRRM is unused.

6.2 Non-CPU Agent Intel® RDT Register Location

The Non-CPU agent Intel RDT related register set (MMIO interfaces) must reside on at least one 4 KB-aligned memory mapped page. The exact location for the register region is implementation-dependent and is communicated to system software by BIOS through the IRDT ACPI structure (see Chapter 5). Multiple RCSes could be mapped to the same 4 KB-aligned page, or distinct pages. No other unrelated registers may be present in the pages used for non-CPU agent Intel RDT. A Virtual Machine Monitor (VMM) or operating system may use page-based access controls to ensure that only designated entities may use the non-CPU agent Intel RDT controls.

When accessing non-CPU agent Intel RDT MMIO interfaces, note that writes to reserved fields, writes to reserved offsets within the MMIO space, or writes of values greater than the supported maximum for a field will be ignored by hardware.

6.2.1 Software Access to Registers

Software interacts with the non-CPU agent Intel RDT features by reading and writing memory-mapped registers. The following requirements are defined for software access to these registers.

- When updating registers through multiple accesses (whether in software or due to hardware disassembly), certain registers may have specific requirements on how the accesses should be ordered for proper behavior. These are documented as part of the respective register descriptions.
- Locked operations to non-CPU agent Intel RDT related registers are not supported. Software should not issue locked operations to non-CPU agent Intel RDT feature hardware registers.

6.2.2 Register Descriptions for Non-CPU Agents

6.2.2.1 Link Interface Type RMID/CLOS Tagging MMIO Interfaces

The IRDT ACPI structures defined in Chapter 4 define MMIO interfaces for configuring the RMID/CLOS for each link interface type, as defined in the RCS structures. An MMIO pointer defined in the RCS fields describes where the configuration interface exists for a particular link interface type. The MMIO locations are specified as absolute physical addresses.

Table 6-3 shows the MMIO field layout for RMID and CLOS tagging, and bandwidth shaping. A common format is used for all RCS types, including for instance RCS instances that support PCIe or CXL use the same field layout.

Common table format across all RCS-Enumerated MMIO.

Table 6-3. MMIO Table Format

Register Name	Mem Offset	Length (B)	Comments
IO RDT Reserved	0x0000	Variable	Reserved
IO_PQR_CLOS0	RCS :: CLOS Block Offset	RCS :: REGW	Common across all RCS types
IO_PQR_CLOS1	IO_PQR_CLOS0 + RCS :: REGW	RCS :: REGW	Per-channel
IO_PQR_CLOS2	IO_PQR_CLOS0 + RCS :: REGW*2	RCS :: REGW	Per-channel
…	Variable	Variable	-
Reserved	Variable	Variable	-
IO_PQR_RMID0	RCS :: RMID Block Offset	RCS :: REGW	Common across all RCS types
IO_PQR_RMID1	IO_PQR_RMID0 + RCS :: REGW	RCS :: REGW	Per-channel
IO_PQR_RMID2	IO_PQR_RMID0 + RCS :: REGW*2	RCS :: REGW	Per-channel
…	Variable	Variable	-
Reserved	Variable	Variable	-
IO_RDT Reserved	Variable	Variable	Remainder of the page

Note that the RCS :: REGW field indicates the register access width of the fields in Table 6-3, either 2B or 8B. Depending on the implementation, this width may be 2 bytes or 8 bytes. The width is indicated by the REGW field in the RCS Table (Section 0).

Note that the base of the RMID and CLOS fields are enumerated in the RCS structure, and the size of these fields varies with the number of supported channels. The set of configurable RMIDs and CLOSs are organized as contiguous blocks of 4B registers.

The “PQR” fields starting at the enumerated offset (RCS :: CLOS Block Offset) are defined with enumerated register field spacing of RCS :: REGW, which may require either 2B or 8B register accesses. A block of CLOS registers exists, followed by a block of RMID registers, indexed per Channel. That is, setting a value in the IO_PQR_CLOS0 field will specify the CLOS to be used for Channel[0] on this RCS.

The valid field width for RMID and CLOS is defined via CPUID leaves (see Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3B for details) for shared-L3 configuration.

Higher offsets allow multiple Channels to be programmed (above Channel 0) if supported. Given that PCIe supports multiple VCs, multiple channels may be supported in the case of PCIe links, but CXL links support only two entries, one at IA_PQR_CLOS0 and one at IO_PQR_RMID0 in this table.

The RMID and CLOS fields are interpreted as numeric tags, exactly as they are in the CPU agent Intel RDT feature set, and software may assign RMID and CLOS values as needed.

Software may reconfigure RMID and CLOS field values at any point during runtime, and values may be read back at any time. As all architectural CPU agent Intel RDT infrastructure, it is dynamically reconfigurable, this enables control loops to work across the capabilities sets collaboratively and consistently.

§

7 Programming Guidelines

7.1 Intel® RDT Monitoring Software Flows for CPU Agents

Intel RDT Monitoring software flows for CPU agents in certain example software implementations are briefly described in this section to provide context for how an end-user could view and use the RDT features. While this chapter provides examples and recommended flows, it is in no way limiting to use models once enumeration and configuration capabilities are enabled in software, and many varied software implementations and usages of RDT beyond the listed examples have been observed.

7.1.1 Intel® RDT Monitoring Software Flows for CPU Agents

Software should first verify the existence of the RDT Monitoring feature(s) before attempting to configure them and read back monitoring data. Periodic management by software may also be required to maintain the proper RMID mapping on a logical thread when context switching or receiving an interrupt for instance (see Section 3.1.1 for details).

7.1.1.1 Step 1 – Enumeration

Before attempting to read or write MSRs associated with the Intel RDT Monitoring feature software should first execute the CPUID instruction and parse its output to ensure that Intel RDT Monitoring and any sub-features to be used (for example, CMT, MBM) are supported on the platform, otherwise General Protection (#GP(0)) faults will be generated.

As discussed in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, if CPUID feature flag CPUID.0x7(Structured Extended Feature Flags).0:EBX[12] is set to '1' then Intel RDT Monitoring is generally supported on the platform.

Once Intel RDT Monitoring support has been verified software should use CPUID.0xF(Shared Resource Monitoring Enumeration leaf).0:EDX to examine which platform resources support monitoring. After the call to CPUID, the EBX register will indicate the maximum RMID supported on the current socket (though particular resources may support fewer RMIDs and this can be enumerated on a per-resource basis as described next).

Software may use CPUID.0xF(Shared Resource Monitoring Enumeration leaf).ResID to determine the number of RMIDs supported for the specific resource in question, the event type bitmask to program into IA32_QM_EVTSEL to retrieve the data for that event in IA32_QM_CTR, and the upscaling factor as discussed in the feature-specific chapters. Software may optionally choose to

build a record of these enumeration responses for each resource to reduce overhead from repeated CPUID calls.

Given that certain processors may support multiple L2 caches, multiple L3 caches, and a variety of logical processor types, it is recommended that software use CPUID from the perspective of each logical processor to comprehend any asymmetric resource support which may be present.

Software should parse Processor Family, Model and Stepping (FMS) to verify that a particular processor includes support for a given model-specific feature. To find out which features are supported on which specific products, refer to Appendix A.3.

7.1.1.2 Step 2 – RMID Association

After verifying that the platform supports Intel RDT Monitoring, software should associate each logical thread or VM of interest with an RMID such that resource utilization by the threads can be tracked. It is expected in general that if an OS or VMM moves an application from one core or socket to another that the RMIDs will be updated (moved along with the app or remapped onto another socket as needed) to maintain an accurate mapping between the applications of interest and the RMIDs programmed onto a logical thread.

Threads by default are initialized to RMID[0], which provides insight into memory bandwidths for the system but not necessarily cache occupancy (which would read 100% occupied in a non-idle system).

7.1.1.3 Step 3 – Event Selection Setup

After associating RMIDs with threads and updating the IA32_PQR_ASSOC register for each thread as needed while running (to account for context swaps and thread migration between cores), software may execute for an arbitrary period of time while hardware tracks occupancy before polling for the resulting occupancy.

After applications have executed for the desired time period software may program an RMID and event code into the IA32_QM_EVTSEL MSR, which will cause the corresponding data to be available in the IA32_QM_CTR MSR (discussed in the following section).

7.1.1.4 Step 4 – Data Sampling

After the IA32_QM_EVTSEL MSR has been programmed with an RMID / Event ID combination the corresponding event data can be read back from the IA32_QM_CTR MSR, which has a bit field layout as defined in Section 3.1.1. Software must check both the Error bit and the Unavailable bit to verify that the data returned is valid (along with the Overflow bit if supported) – if an error is indicated the monitoring data reported back must not be used.

As described in Section 3.1.1 the Error bit will be set if an RMID greater than the global maximum (specified in CPUID) is programmed into IA32_QM_EVTSEL, or an unknown/unsupported Event ID is programmed.

Similarly, the Unavailable bit is set when data is requested for an RMID that does not support that particular resource or does not support an RMID value that high.

An example is if occupancy monitoring of resource "A" supported four RMIDs, and resource "B" supported 2 RMIDs. If software requested the occupancy of either Resource A or B for RMIDs 0 or 1 then valid data would be reported back. If occupancy data for RMIDs 2 or 3 was requested for resource "B" however data would not be reported, and the Unavailable bit would be set.

The Overflow bit, if supported, is set when an overflow of an incrementing counter is triggered, allowing software to correct or discard errant values that may lead to erroneous bandwidth calculations.

If an error is indicated, it will be cleared automatically once valid values are programmed into IA32_QM_EVTSEL and any hardware conditions preventing accurate monitoring are resolved. The Overflow bit, if implemented, is cleared on a read of IA32_QM_CTR.

7.1.1.5 Step 5 – Sample CMT/MBM Data Collection and Analysis

Once CMT and MBM data has been collected it can be interpreted as described in the following example.

Consider the case where CMT and MBM are supported on a platform, and a large number of RMIDs are available. On this platform the user seeks to profile two threads within an application, so both threads are assigned individual RMIDs and run on separate physical cores for a period of one second, then occupancy and bandwidths are read back via the MSR interface (IA32_QM_EVTSEL and IA32_QM_CTR). In this example, the following parameters are key to interpreting the results:

- System topology – two Intel® Xeon™ CPUs with 14 cores per socket, and a 3-level cache subsystem, where the last-level cache totals 35 MB per socket.
- The last-level cache is verified using CPUID leaf 0x4 as the last level cache between the cores and memory, meaning L3 external bandwidth values can be used to measure memory bandwidth.
- As enumerated via CPUID the upscaling factor (CPUID.0xF(Shared Resource Monitoring Enumeration leaf).1:EBX) to convert counter values to final values in bytes is 0xE000 (decimal 57344).
- Since the total L3 cache size is 36700160 bytes and the upscaling factor is 57344, we know that the maximum possible CMT occupancy counter value reported by the system will be total cache size divided by the conversion factor, or 36700160/57344 = 640.
    - As the threads are profiled, we can compare the reported occupancy to the maximum occupancy counter value, giving an indication of what fraction of the total cache an application is using without needing to convert to bytes first.

Suppose that the threads are configured as follows:

- Associate thread[0] with RMID[1].
- Associate thread[1] with RMID[2].
- Leave all other threads in the system with the default RMID[0] association.

In order to profile memory bandwidth an initial sampling of the free-running MBM counters is required:

- Program IA32_QM_EVTSEL with RMID[1] for the first thread and event code 0x2 for total L3 external bandwidth, then read the corresponding data from IA32_QM_CTR (and verify that the Unavailable and Error bits in IA32_QM_CTR are not set so the data is valid).
- Program IA32_QM_EVTSEL with RMID[1] for the first thread and event code 0x3 for local L3 external bandwidth, then read the corresponding data from IA32_QM_CTR (and verify that the Unavailable and Error bits in IA32_QM_CTR are not set so the data is valid).
- Repeat these steps with RMID[2] for the second thread.

Note that we assume that RMID[1] and RMID[2] have previously been used for profiling other applications, so they may initially contain nonzero occupancy and bandwidth counter values.

Note that in this example we assume that RMID[1] and RMID[2] are set up exclusively for the use of the two threads being profiled, and that these threads are not currently scheduled, and they have no data in the L3 cache, so the bandwidth counters, even if they contain initial values, are not changing. The occupancy counters may change even if no threads are scheduled using RMID[1] and RMID[2] however if they have previously run and have data in the L3 cache as other threads on the system run and cache space is dynamically redistributed due to evictions and standard cache LRU policies.

Note that if the threads in RMID[1] and RMID[2] are running while we measure initial counter values then skew may appear in the counter values, proportional to the time delay between reading each of the event codes (which should be minimized) and the bandwidths consumed by the application (which may vary significantly based on application behavior).

Now that initial MBM counter values have been established, the program can be left to run for a period of time, in this case one second. The Intel RDT Monitoring data can then be read back as follows:

- Program IA32_QM_EVTSEL with RMID[1] for the first thread and event code 0x1 for L3 cache occupancy, then read the corresponding data from IA32_QM_CTR (and verify that the Unavailable and Error bits in IA32_QM_CTR are not set so the data is valid).
- Program IA32_QM_EVTSEL with RMID[1] and the event code for total L3 external bandwidth (0x2), read the data from IA32_QM_CTR and again verify that the “U” and “E” bits are not set.
- Similarly read back local L3 external bandwidth using the event code 0x3 and verify that the data is valid.

- Repeat the previous three steps with RMID[1] to read back the Intel RDT monitoring metrics for the second thread.

Example data read back after profiling for one second is shown in the following table.

Table 7-1. Example CMT and MBM Counter Values

	Thread 0		Thread 1	
Event Type	First Sample	Second Sample	First Sample	Second Sample
L3 Cache Occupancy	N/A	0x25	N/A	0x180
Total L3 External Bandwidth	0x00FE985E	0x00FEBC14	0x00002541	0x0000D9F7
Local L3 External Bandwidth	0x0A8C9512	0x0A8CB5ED	0x00000314	0x0000AC5D

Note that in the previous sample data the counter values are shown as 32-bit values, implying that the upper fields in the counter MSR were either zeroes or not changing and can be disregarded – this may not always be the case however when bandwidths are high, or in the case of future counters which may increment quickly.

In the example, the final cache occupancy for the threads can be calculated as follows:

- Thread[0]: CounterValue * UpscalingFactor = 37*57344 = 2121728 bytes (roughly 2.02 MB).
- Thread[1]: CounterValue * UpscalingFactor = 22020096 bytes = 21 MB.

Thus, based on the CMT profiling of the two example threads, we see that Thread[0] consumes around 2MB of cache space, and Thread[1] consumes around 21MB, over 10x more, which indicates that it likely has a larger data working set or it may be partly streaming through memory. Software should also consider memory bandwidth readings to determine whether Thread[1] is simply cache-friendly or whether it is a streaming application.

Total memory bandwidth values for the two threads can be determined as follows:

- Thread[0]: (Second counter reading – First counter reading)*UpscalingFactor = (0x00FEBC14-0x00FE985E)*57344 = 9142*57344 = 524238848 bytes/second, or around 500 MB/s since we sampled for one second.
- Thread[1]: (Second counter reading – First counter reading)*UpscalingFactor = (0x0000D9F7-0x00002541)*57344 = 46262*57344 = 2652848128 bytes/second, or around 2.5 GB/s.

Local memory bandwidth values for the two threads can be determined as follows:

- Thread[0]: (Second counter reading – First counter reading)*UpscalingFactor = (0x0A8CB5ED-0x0A8C9512)*57344 = 8411*57344 = 482320384 bytes/second, or around 460 MB/s.

- Thread[1]: (Second counter reading – First counter reading)*UpscalingFactor = (0x0000AC5D-0x00000314)*57344 = 43337*57344 = 2485116928 bytes/second, or around 2.3 GB/s.

Based on the prior calculations we observe that Thread[0] has low memory bandwidth demands at roughly 500 MB/s, and Thread[1] uses more bandwidth at 2.5 GB/s, but not enough to classify it as a streaming thread. With its 21 MB cache occupancy and moderate memory bandwidth, Thread[1] is best classified as a cache-friendly thread, though observing its behavior over a longer period of time and sampling other system metrics to better understand its time-variant behavior and compute requirements is recommended if detailed profiling is the goal.

Note that in this example most of the bandwidth demands of the threads are satisfied by the memory controller on the local CPU, meaning bandwidth associated with the QPI link and other sources is low, implying that the NUMA-aware OS properly located the memory allocation for the threads on the same socket as the running threads.

This may not always be the case however, and if a bandwidth imbalance is detected then we may choose to either move the compute threads to the other CPU (closer to the data in memory) or move the data in memory to another address range within the scope of the local CPU memory controller for better performance.

7.1.2 Native OS Environments

In a non-virtualized environment, the RMIDs can be associated with applications or application threads. The OS may even choose to associate different parts of a single application to be associated with different RMIDs if needed. But a typical usage would save and restore the RMIDs along with the context information during the context switch.

For multi-threaded applications, multiple threads can share the same RMID. The implications stated earlier also apply to multi-threaded applications with the following additional considerations for shared code/data. For example, if app0 was multi-threaded (for example, two threads per application), then we can get occupancy information for each thread of application. The only additional implication here is that the occupancy of the threads that share data will be associated to the thread that filled the shared data. Heuristics that minimize contention in the shared cache for single threaded workloads to optimize total system throughput and to provide QoS will also be effective for the multi-threaded workloads.

7.1.3 Virtualization Scenarios

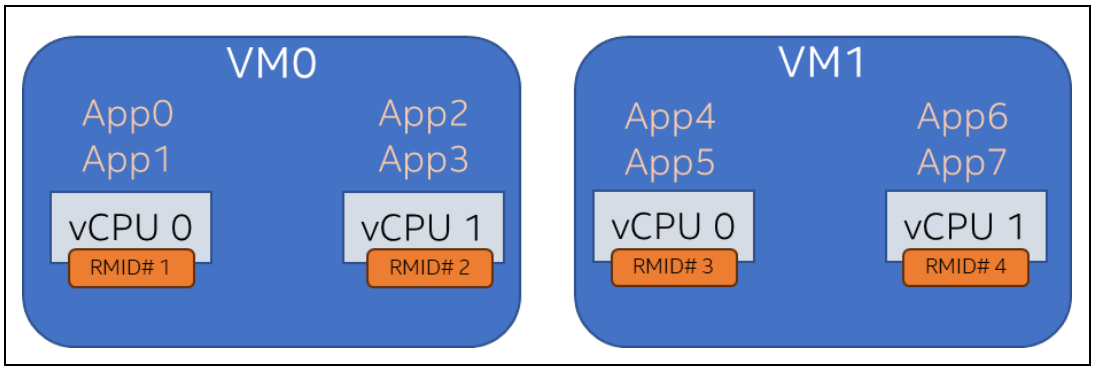
In case of virtualization, RMIDs can be allocated in different ways. The VMM can choose to allocate the RMIDs to different VMs or vCPUs. The current planned implementations do not support reporting individual occupancies of applications running within a VM unless the VMM and guest OS are both

enabled to support Intel RDT. The RMID assignment at VM and VCPU level are described next.

RMIDs assigned to Virtual Machines (VMs): In this usage case RMIDs are assigned to VMs instead of applications and the occupancy reported is on a per VM basis. Multiple applications running within a VM will have a consolidated occupancy which will be reported by the RMID. Profiling of workloads and heuristics that optimize for overall system throughput and for providing QoS based on SLAs would be based in the granularity of VMs. Hence to provide QoS, HP applications can be mapped to a VM with a high priority so that scheduling decisions to minimize contention will treat all applications running in the HP VM as high priority. The heuristics that work on occupancy monitoring based on contention in the shared cache will still be effective but will work in the granularity of VMs. When scheduling VMs, the VMM can use the occupancy monitoring information available for the VMs from the RMIDs. There are no other additional implications for VMs.

RMIDs assigned to vCPUs within VMs: In this usage case scenario, RMIDs are assigned to vCPUs within a VM. Since there maybe multiple applications within a VM running on the vCPUs, the occupancy reported by the RMID for a vCPU will represent the consolidated occupancy of the applications running on that vCPU. As an example, if there are two VMs with 2 vCPUs each and there are four applications in each VM as shown in Figure 7-1.

Figure 7-1. RMIDs Assigned to vCPUs



The occupancy reported by the RMID assigned to vCPU0 will represent the consolidated occupancy of App0 and App1. Similarly, only the consolidated occupancy of App2 and App3 is what will be reported and so on. Hence optimizations for system throughput, QoS and application profiling would have to be at the granularity of vCPUs. The OS running within a VM will have its own scheduling policy that would determine how applications are scheduled to the vCPUs.

When applications migrate within a VM from one vCPU to another, the consolidated occupancy reported will also be affected as it would depend on the nature of the applications scheduled to a vCPU. Hence any policy or heuristic that is implemented should be in the granularity of VCPU profiling. The recommended approach is to profile the workload at a vCPU level and then design heuristics based on vCPU profiles to optimize for throughput and provide QoS.

7.2 Intel® RDT Allocation Software Flows for CPU Agents

RDT Allocation software flows for CPU agents are briefly described in this section to provide context for how and end-user may view the feature.

7.2.1 Intel® RDT Software Allocation Flows for CPU Agents

7.2.1.1 Step 1 – Enumeration

Before attempting to read or write MSRs associated with the Intel RDT Allocation feature software should first poll CPUID to ensure that Intel RDT Allocation and any sub-features to be used (for example, L3 CAT, L2 CAT, MBA) are supported on the platform, otherwise General Protection (#GP(0)) faults will be generated. As discussed in Section 3.2, if CPUID feature flag CPUID.0x7(Structured Extended Feature Flags).0:EBX[15] is set to '1' then Intel RDT Allocation is generally supported on the platform.

Once Intel RDT Allocation support has been verified software should poll and examine CPUID.0x10.0:EBX to examine which platform resources support allocation. After the call to CPUID, the EBX register will indicate the supported Intel RDT Allocation features on the current socket.

Software may use CPUID.0x10.ResID to determine the number of CLOS supported for the specific resource in question, the max length of the CAT bitmask, the max MBA delay value, and so on, and presence of sub-features like CDP on top of CAT for a given level of the cache. Software may optionally choose to build a record of these enumeration responses for each resource to reduce overhead from repeated CPUID calls.

Software should parse Processor Family, Model and Stepping (FMS) to verify that a particular processor includes support for a given model-specific feature. To find out which features are supported on which specific products, refer to Appendix A.3.

Note that it is important that software enumerate the Intel RDT Monitoring capabilities of the platform in the order specified in Section 3.1.1.

7.2.1.2 Step 2 – Optionally Enable CDP

If software wants to use CDP, enable it via the IA32_PQOS_CFG MSR.

7.2.1.3 Step 3 – Mask and Bandwidth Control Setup

After determining the presence of hardware Intel RDT Allocation support software should configure the CAT masks and MBA delay values if supported to provide capacity allocation/bandwidth hints to the hardware via the IA32_ResourceType_QOS_MASK_n MSRs and IA32_L2_QOS_Ext_BW_Thrtl_n

MSRs, depending on the usage model specified in Section 3.1.1 and the number of CLOS available (enumerated in feature-specific ResID sub-leaves).

It is considered good practice to first verify that IA32_L3_QOS_MASK_0 contains all "1" to the length of the bitmask (such that CLOS0 can access the entire cache) and that all threads are in CLOS0 before making changes to the masks (which may otherwise result in rapidly changing cache available to applications, which may lead to performance variation, though no functional errors are possible). Also verify that no bandwidth enforcement is configured in the IA32_L2_QOS_Ext_BW_Thrtl_n MSRs. It is also considered best practice to set up CLOS[0] as the highest priority CLOS with a large fraction of the cache, CLOS1 as the next highest, and so on.

7.2.1.4 Step 4 – CLOS Association

After the CAT/CDP per-CLOS mask MSRs are set up to known values, whether overlapped, shared or a combination depending on application needs and goals, and after MBA delay values are set up, each of the threads should be associated into a desired Class of Service via the IA32_PQR_ASSOC MSR. This MSR may be read or written at any time.

As part of some implementations an OS may choose to set up masks then change the IA32_PQR_ASSOC MSR on context switches (to associate a portion of the cache with an application or thread for instance).

7.3 Intel® RDT Software Flows for Non-CPU Agents

This section describes software architecture considerations for Intel RDT features for non-CPU agents, recommended usage flows and related considerations. This builds upon the architectural concepts and software usage examples discussed in Chapter 4.

Software seeking to use RDT for non-CPU agents has a number of tasks to comprehend:

- Enumeration of the capabilities of Intel RDT for CPU agents (through CPUID) and Intel RDT for non-CPU agents (through CPUID and ACPI).
- Reservation of (or comprehension of the sharing implications of using) RMIDs and CLOS from the pools available at each resource level and subject to the RMID and CLOS management best practices on a particular processor.
- Pre-configuration of any resource limits to be used for modulating device activity, such as a cache mask for a CLOS intended to be used with a device.
- Configuration of each device's tagging properties through the MMIO interface described by the ACPI structures, such as associating a device with a particular RMID, CLOS and bandwidth limit, as applicable.

- Enabling the Intel RDT features for non-CPU agents through the enable MSR infrastructure -- the IA32_L3_IO_QoS_CFG MSR is shown in Figure 4-2, at MSR address 0xC83.
- Periodically adjusting resource limits subject to software policies and any control loops which may be present.
- Comprehending the implications of Sub-NUMA clustering (SNC) if present and enabled.

§

A Intel® RDT Feature Details

A.1 Intel® RDT Feature Evolution

This section describes various generations of product and Intel RDT feature intercepts. Intel RDT provides a number of monitoring and control capabilities for shared resources in multiprocessor systems. This section covers updates to the feature that are available in current and future Intel processors, starting with brief descriptions followed by tables with details.

1. **Intel® RDT on the 3rd Gen Intel® Xeon® Scalable Processor Family.** The 3rd Gen Intel® Xeon® Scalable Processor Family, based on Ice Lake server microarchitecture, adds the following Intel RDT enhancements:
   - 32-bit MBM counters (versus 24-bit in prior generations), and new CPUID enumeration capabilities for counter width.
   - Second generation Memory Bandwidth Allocation (MBA): Introduces an advanced hardware feedback controller that operates at microsecond timescales, and software-selectable min/max throttling value resolution capabilities. Baseline descriptions of the MBA "throttling values" applied to the threads running on a core are described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.
     Second generation MBA capabilities also add a work-conserving feature in which applications that frequently access the L3 cache may be throttled by a lesser amount until they exceed the user-specified memory bandwidth usage threshold, enhancing system throughput and efficiency, in addition to adding more precise calibration and controls. Certain BIOS implementations may further aid flexibility by providing selectable calibration profiles for various usages.
   - 15 MBA / L3 CAT CLOS: Improved feature consistency and interface flexibility. The previous generation of processors supported 16 L3 CAT Class of Service tags (CLOS), but only 8 MBA CLOS. The changes in enumerated CLOS counts per-feature are enumerated in the processor as before, via CPUID.
2. **Intel® RDT on Intel Atom® Processors, Including the P5000 Series.** Intel Atom® processors, such as the P5000 series, based on Tremont microarchitecture add the following Intel RDT enhancements:
   - L2 CAT/CDP: L2 CAT/CDP and L3 CAT/CDP may be enabled simultaneously on supported processors. As these are existing features defined in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, no new software enabling should be required.
   - Supported processors match the capabilities of the 3rd Gen Intel Xeon Scalable Processor Family based on Ice Lake Server microarchitecture, including traditional Intel RDT uncore features: L3 CAT/CDP, CMT, MBM, and second-generation MBA. As these features are architectural, no new software enabling is required. Related enhancements in Intel Xeon processors also carry forward to

supported Intel Atom processors, with consistent software enabling. These features include 32-bit MBM counters, second generation MBA, and 15 MBA/L3 CAT CLOS.

3. **Intel® RDT in processors based on the 4th Gen Intel® Xeon® Scalable Processor Family.**
   Processors based on 4th Gen Intel® Xeon® Scalable Processor Family add the following Intel RDT enhancements:
   — STLB QoS: Model-specific capability to manage the second-level translation lookaside buffer structure within the core (STLB) in a manner quite similar to CAT (CLOS-based, with capacity masks). This may enable software that is sensitive to TLB performance to achieve better determinism. This is a model-specific feature due to the microarchitectural nature of the STLB structure. The code regions of interest should be manually accessed.

4. **Intel® RDT in Processors Based on 5th Gen Intel® Xeon® Processors.**
   Processors based on 5th Gen Intel® Xeon® Processors add the following Intel RDT enhancements:
   — L2 CAT and CDP: Includes control over the L2 cache and the ability to partition the L2 cache into separate code and data virtual caches. No new software enabling is required; this is the same architectural feature described in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.

5. **Future Intel® RDT.**
   Future processors add the following Intel RDT enhancements:
   — Third generation Memory Bandwidth Allocation (MBA): new per-logical-processor capability for bandwidth control (rather than the more coarse-grained core-level throttling value resolution in prior generations). This capability enables more precise bandwidth shaping and noisy neighbor control. Some portions of the control infrastructure now operate at core frequencies for controls that are responsive at the nanosecond level.
   — Intel® RDT features support for non-CPU agents, enabling advanced monitoring and control capabilities for PCIe and CXL devices, as well as integrated processor accelerators.
   — Region Aware Memory Bandwidth Monitoring (MBM) and Region Aware Memory Bandwidth Allocation (MBA).

A.2 Intel® RDT Architectural Features and Supported Products

	Intel RDT Feature Category	Shared Resource	Agent	Intel RDT Sub-Feature	Intel RDT Scope	Supported Products
Monitoring	Cache Monitoring Technology (CMT)	L3	CPU	L3 CMT for CPU agents	Per-thread RMID-based	Intel® Xeon® E5/E7 v3,v4, Intel® Xeon® D, Intel® Xeon® Scalable Processor, 2nd Gen Intel® Xeon® Scalable Processor, 3rd Gen Intel® Xeon® Scalable Processor, 4th Gen Intel® Xeon® Scalable processor, 5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series, Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
		L3	I/O	L3 CMT for non-CPU agents	Per-agent RMID-based	Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
	Memory Bandwidth Monitoring (MBM)	-	CPU	MBM Local for CPU agents	Per-thread RMID-based	Intel® Xeon® E5/E7 v4, Intel® Xeon® D, Intel® Xeon® Scalable Processor, 2nd Gen Intel® Xeon® Scalable Processor, 3rd Gen Intel® Xeon® Scalable Processor, 4th Gen Intel® Xeon® Scalable processor, 5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series, Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
			CPU	MBM Total for CPU agents	Per-thread RMID-based	Intel® Xeon® E5/E7 v4, Intel® Xeon® D, Intel® Xeon® Scalable Processor, 2nd Gen Intel® Xeon® Scalable Processor, 3rd Gen Intel® Xeon® Scalable Processor, 4th Gen Intel® Xeon® Scalable processor, 5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series (Selected Processors), Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)

	Intel RDT Feature Category	Shared Resource	Agent	Intel RDT Sub-Feature	Intel RDT Scope	Supported Products
			CPU	Region Aware MBM for CPU agents	Per-RMID and per-Region based	Future Intel® Processors
			I/O	MBM Local for non-CPU agents	Per-agent RMID-based	Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
			I/O	MBM Total for non-CPU agents	Per-agent RMID-based	Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)

	Intel RDT Feature Category	Shared Resource	Agent	Intel RDT Sub-Feature	Intel RDT Scope	Supported Products
Allocation	Cache Allocation Technology (CAT)	L2	CPU	L2 CAT for CPU agents	Per-thread CLOS-based	Atom Server C3000, 5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series, Intel® Xeon® Scalable processor (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
				L2 CDP for CPU agents	Per-thread CLOS-based	5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series, Intel® Xeon® Scalable processor(codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
		L3	CPU	L3 CAT for CPU agents	Per-thread CLOS-based	Intel Atom® X Series (Selected Processors), Intel® Xeon® E5/E7 v3 (Selected Processors), Intel® Xeon® E5/E7 v4 , Intel® Xeon® D, Intel® Xeon® Scalable, 2nd Gen Intel® Xeon® Scalable Processor, 3rd Gen Intel® Xeon® Scalable Processor, 4th Gen Intel® Xeon® Scalable processor, 5th Gen Intel® Xeon® Scalable processor, Intel® Xeon® W, Intel Atom® Processor P5000 Series, Intel® Xeon® Scalable processor(codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
				L3 CDP for CPU agents	Per-thread CLOS-based	Intel® Xeon® E5/E7 v4, Intel® Xeon® Scalable Processor, 2nd Gen Intel® Xeon® Scalable Processor, 3rd Gen Intel® Xeon® Scalable Processor, 5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series, Intel® Xeon® Scalable processor(codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)
			I/O	L3 CAT for non-CPU agents	Per-agent, CLOS-based	Intel® Xeon® processors (codename Granite Rapids), Intel® Xeon® processors (codename Sierra Forest)

	Intel RDT Feature Category	Shared Resource	Agent	Intel RDT Sub-Feature	Intel RDT Scope	Supported Products
	Memory Bandwidth Allocation (MBA)	L3 external bandwidth	CPU	MBA for CPU agents (First Generation MBA)	Per-interface, CLOS-based	Intel® Xeon® Scalable Processor, 2nd Gen Intel® Xeon® Scalable Processor, 3rd Gen Intel® Xeon® Scalable Processor formerly codenamed *Cooper Lake*
			CPU	MBA for CPU agents (Second Generation MBA)	Per-interface, CLOS-based	3rd Gen Intel® Xeon® Scalable Processor (excluding codename *Cooper Lake*), 4th Gen Intel® Xeon® Scalable processor, 5th Gen Intel® Xeon® Scalable processor, Intel Atom® Processor P5000 Series, Future Intel® Xeon® Scalable processor (codename *Granite Rapids*), Future Intel® Xeon® processors (codename *Sierra Forest*)
			CPU	MBA for CPU agents (Third Generation MBA)	Per-agent, CLOS-based	Future Intel® Xeon® processors (codename *Granite Rapids*), Future Intel® Xeon® processors (codename *Sierra Forest*)
			CPU	Region Aware MBA for CPU Agents	Per-agent, CLOS-based and per-Region based	Future Intel® Processors
	Cache Bandwidth Allocation (CBA)	-	CPU	CBA for CPU agents	Per-Logical Processor based	Future Intel® Processors

A.3 Intel® RDT Model-Specific Features and Supported Products

Intel RDT Feature Category	Supported Products
Resource Aware MBA (MBA4.0)	• Intel® Xeon® processors (codename Granite Rapids). • Intel® Xeon® processors (codename Sierra Forest).
Intel® RDT and Sub-NUMA Clustering (SNC) Compatibility	• 3rd Gen Intel® Xeon® processors. • 4th Gen Intel® Xeon® processors. • 5th Gen Intel® Xeon® processors.
STLB QoS	4th Gen Intel® Xeon® processors. The following product generations on SKUs with Intel® Time Coordinated Computing (Intel® TCC) support: • 11th Gen Intel® Core™ Processors (UP3-Series). • Intel® Xeon® W Processors (TGL-H).

Intel RDT Feature Category	Supported Products
	• 12th Gen Intel® Core™ Processors (S-Series). • 13th Gen Intel® Core™ Processors (P-Series). • 13 Gen Intel® Core™ Processors (S-Series). • Intel Atom® x7000E Series Processors.

A.4 Feature Mapping: CPU Agents, Non-CPU Agents in Different L3 Configurations

Configuration	Intel RDT Feature	CPU Agents Intel RDT Scope	Non-CPU Agents Intel RDT Scope	Comments
Shared-L3	Cache Monitoring Technology (CMT)	Per-thread RMID-based	Per-agent RMID-based	Unified per-RMID counters across CPU Agents and non-CPU Agents.
Shared-L3	Memory Bandwidth Monitoring (MBM)	Per-thread RMID-based	Per-agent RMID-based	Unified per-RMID counters across CPU Agents and non-CPU Agents.
Shared-L3	Cache Allocation Technology (CAT)	Per-thread CLOS-based	Per-agent CLOS-based	Unified per-CLOS controls across CPU Agents and non-CPU Agents.
Shared-L3	Code and Data Prioritization (CDP)	Per-thread CLOS-based	N/A	CDP is not supported for non-CPU Agents.
Shared-L3	Memory Bandwidth Allocation (MBA)	Per-agent MBA throttling (MBA3.0 and higher) or Per-interface MBA throttling (MBA1.0-2.0)	N/A	MBA is not supported for non-CPU Agents.

A.5 Architectural MSRs used with Intel® RDT Features

The following architectural Model-Specific Registers are used with Intel® RDT features.

MSR Name	Comments
IA32_PQR_ASSOC	Set the RMID and CLOS pair.
IA32_QM_EVTSEL	Set event codes and RMID to be monitored.
IA32_QM_CTR	Reports monitoring telemetry data.
IA32_L3_MASK_n	Bitmask to assign L3 cache ways for each CLOS. "n" registers, one register per CLOS.
IA32_L2_QoS_Ext_BW_Thrtl_n	Set valid throttling levels. "n" registers, one register per CLOS
IA32_L2_QOS_MASK_n	Bitmask to assign L2 cache ways for each CLOS. "n" registers, one register per CLOS.
IA32_L3_IO_QOS_CFG	Set to enable Allocation and Monitoring for non-CPU Agents
IA32_QoS_Core_BW_Thrtl_n	Set valid throttling levels, one byte per CLOS. "n = 0 to (((CLOS_MAX+1)/8) -1)" registers

A.6 Model-Specific Registers for Intel® RDT Model Specific Features

The following notable non-architectural Model-Specific Registers are used with Intel® RDT features and will be expanded over time. Others are discussed in preceding model-specific chapters.

MSR Name	Comments
MBA_CFG	Set the RMID and CLOS pair.
RMID_SNC_CONFIG	Clear to enable RMID Sharing Mode.
STLB_QOS_INFO	Discover STLB QOS parameters
STLB_QOS_MASK_N	STLB QOS Capacity Bitmasks
STLB_FILL_TRANSLATION	Fill a logical address into the STLB
PQR_ASSOC	Resource Association Register
L3_QOS_MASK_N	L3 Class of Service Mask

§

B Model-Specific Intel® RDT Features

B.1 Model-Specific Intel® RDT Features for CPU Agents

This section gives an overview of non-architectural features that are implemented on specific products. To find out which features are supported on which specific products, refer to Appendix A.3.

In certain cases, model-specific features may be implemented rather than architectural features in cases where the cache or memory hierarchies are rapidly evolving, or in cases where usages are specialized and require intricate software enabling and tuning, or in cases where a subset of special-purpose processors are enabled with certain features within a broader product line.

Support for a certain model-specific feature in a particular product generation does not imply that future products will support the same model-specific feature; furthermore, this does not guarantee software forward-compatibility. Software should use Processor Family, Model and Stepping (FMS) to verify that a particular processor includes support for a given model-specific feature.

B.1.1 Resource Aware MBA

Resource Aware MBA (MBA 4.0) for CPU-agent was formerly known as Fourth Generation MBA (MBA 4.0) which supports over Third Generation MBA capabilities as Bandwidth management support is implemented to support up to three different resources – DDR Memory, CXL links, and UPI Links on a pre thread basis. Third generation MBA capabilities (see Section 3.2.3.3) are the default mode of operation, with Resource Aware MBA being opt-in. See Appendix A.3 for Resource Aware MBA feature intercept details.

B.1.1.1 Overview

Resource Aware MBA allows per-thread tracking and control of Bandwidth to different resources – that is, enabling bandwidth control per-thread and per-resource simultaneously. As in the third generation of MBA, each resource and thread are managed by a hardware controller which modulates the bandwidth of each thread targeting a particular downstream resource around a bandwidth target set by Intel RDT software interfaces.

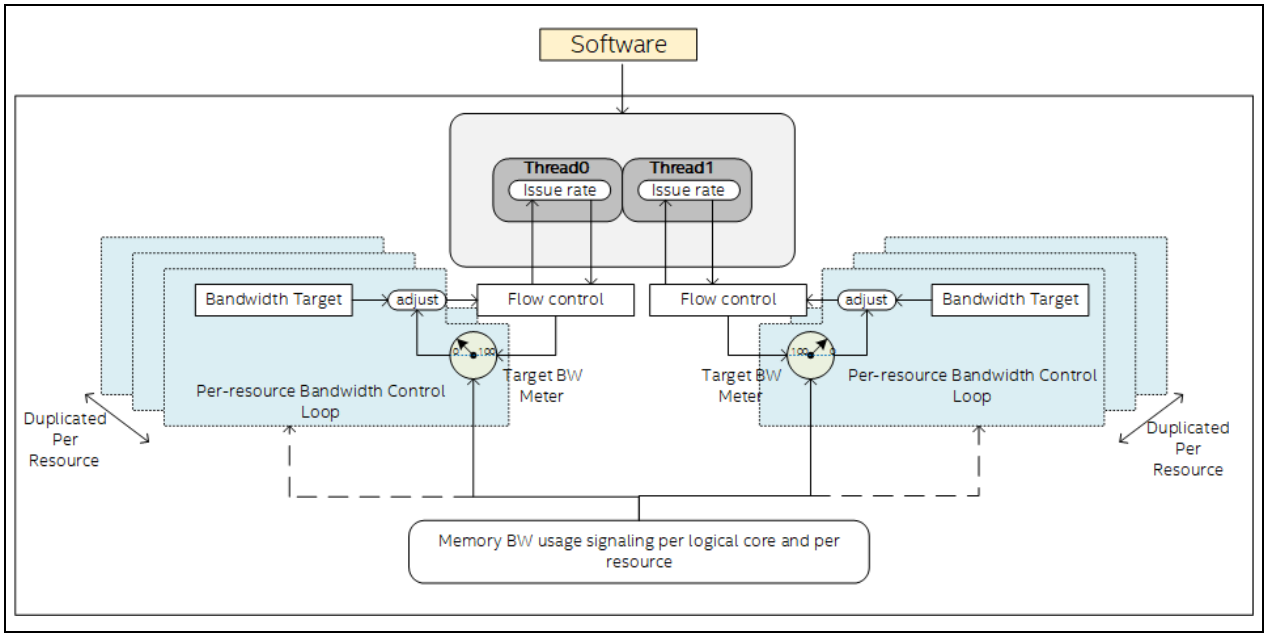
The resource types that are managed are:

1. DDR – All traffic towards DDR Memory regardless of location of location (local, remote or CXL).
2. CXL – All traffic towards CXL resources such as CXL.mem pools including remote.

3. UPI - All traffic that utilizes the Intel® Ultra Path Interconnect (Intel® UPI) link(s) for cross socket data transfer regardless of target on the remote socket.

The high-level implementation of Resource Aware MBA is shown in Figure 7-2.

Figure 7-2. High-Level Overview of the Resource Aware MBA (MBA 4.0)



B.1.1.2 Enable MSR

Resource Aware MBA (MBA 4.0) is opt-in feature. Before configuring MBA throttling values per-thread and per-resource, the feature should be enabled (through a configuration MSR). The MBA_CFG MSR is used to enable the Resource Aware MBA feature for CPU agents.

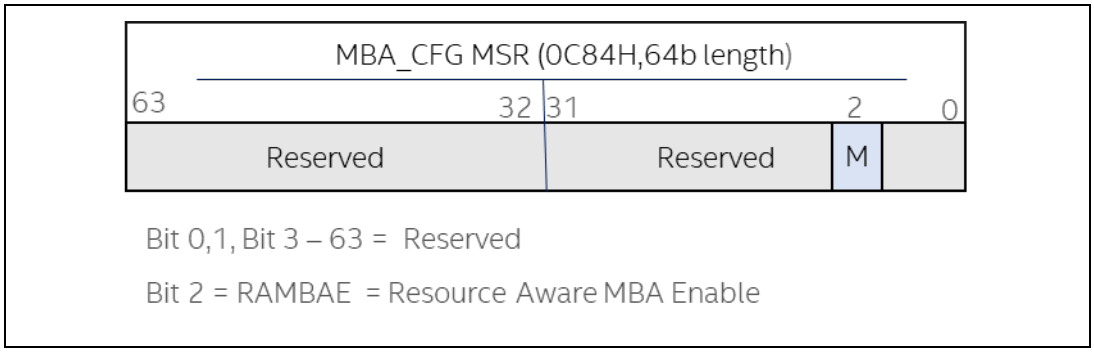
One bit is defined in this MBA_CFG MSR, bit[2], which when set enables the Resource Aware MBA feature and switches between third-generation MBA and Resource Aware MBA modes.

The default value is 0x0 (Resource Aware MBA is disabled by default), and all bits not defined are reserved. Any writes to reserved bits will generate a General Protection Fault (#GP(0)).

This MSR is scoped at the die level and is cleared on system reset. It is expected that software will configure this MSR consistently across all L3 caches that may be present in the SoC.

The definition of the MBA_CFG MSR is shown in Figure 7-3, and its MSR address is 0xC84.

Figure 7-3. The MBA_CFG MSR for Enabling Resource Aware MBA Feature



Reference BIOS implementations supporting Resource Aware MBA will extend the legacy bandwidth profile knobs from Second Generation MBA with a drop-down menu of three options (see Section 5.2 for details)

B.1.2 Intel® RDT and Sub-NUMA Clustering Compatibility

The following sub-sections describe Intel RDT and Sub-NUMA Clustering (SNC) compatibility enabling components. Utilizing SNC and RDT simultaneously may provide resource contention isolation benefits but requires incremental software enabling with the introduction of SNC.

B.1.2.1 Introduction

Following sub-sections describe Intel RDT monitoring features behavior in the presence of either multiple NUMA domains per socket, other product implementations in which multiple NUMA domains may appear per processor, due to either logical or physical resource partitioning. This section references Intel RDT features such as MBA, MBM, CMT and CAT for CPU agents and non-CPU agents described in Chapter 3 and Chapter 4 respectively.

The Sub-NUMA Clustering (SNC) feature creates localization domains within a processor by mapping addresses from a local memory controller to a subset of the L3 slices that are at a reduced distance to nearby memory controller(s), reducing latency, and increasing equivalent traffic isolation across memory channels controllers.

MBA usage is not affected in presence SNC; bandwidth targets apply globally across all SNC domains. L3 CAT and Monitoring features (L3 CMT and MBM) usage is affected in the presence of SNC. Following sections provide details. See Appendix A.3 for Intel RDT and Sub-Numa Clustering (SNC) Compatibility feature supported product details (for example, products where the features are simultaneously supported).

B.1.2.2 SNC Enabled and L3 Cache Allocation Technology

L3 Cache Allocation Technology (L3 CAT) allows an Operating System (OS), Hypervisor / Virtual Machine Manager (VMM) or similar system service management agent to specify the amount of L3 cache capacity of the Resource Allocation Domain (RAD) into which an application can fill.

In the presence of SNC, cache capacity bitmasks are still die-scoped and apply across multiple-L3 domains. Each bit in the cache capacity bitmask manages all clusters and dictates the portion of each SNC cluster available for a given Resource Management Domain. For example, each bit in cache capacity bitmask represents half as much L3 cache capacity at each cluster when SNC2 is enabled, or one-quarter as much L3 cache capacity at each cluster when SNC4 is enabled and so on. Note that total L3 cache capacity does not change.

Software may choose to apply consistent policies across SNC domains utilizing this property, such as CLOS[0] having full access to the cache across any SNC domain in which it may run, but CLOS[1] having access to only half of the cache, implying that it contains a set of lower-priority threads.

B.1.2.3 SNC Enabled and RMID Distribution Modes

There are two modes available to control Resource Monitoring ID (RMID) distribution when SNC is enabled: Default mode and RMID Sharing.

Software should consider and select the mode in which RMIDs are distributed or shared across the SoC and SNC domains depending on its usage needs.

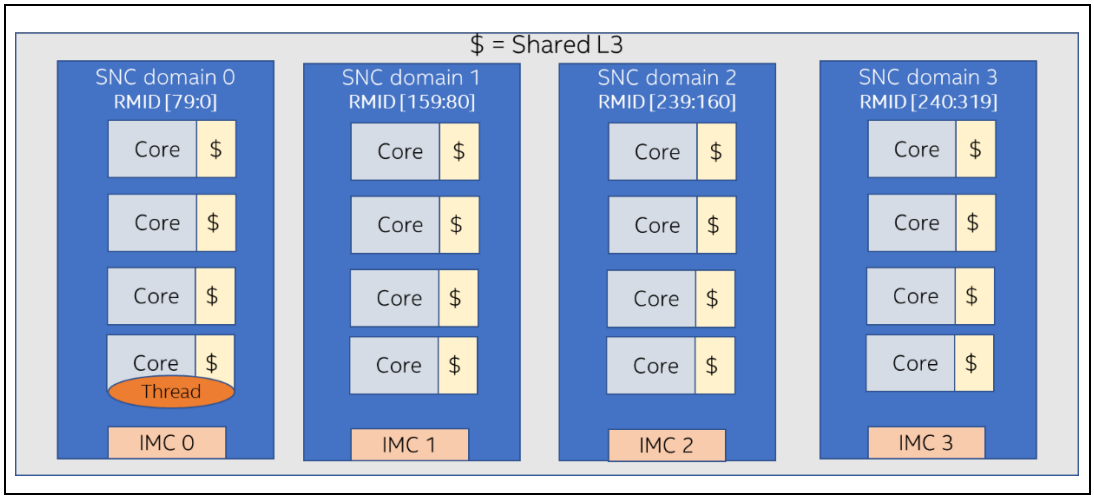
B.1.2.3.1 Default Mode

When SNC is enabled the available pool of RMIDS are distributed across all the L3 slices. RMIDs are distributed across the cores in the same fashion as done when SNC is not enabled, see Figure 7-4.

This distribution scheme allows the RMIDS enumerated by CPUID to be directly used. Software should be aware of the distribution of RMIDs between the SNC domains. For instance, if there are 320 RMIDs available (enumerated via CPUID.(EAX=0FH(Shared Resource Monitoring Enumeration leaf), ECX=0H)) and an SNC-4 configuration is selected, four localization domains exist within a processor.

These 320 RMIDs can be dived into four groups of 80 RMIDS with first 80 allocated to SNC domain 0, the next 80 to SNC domain 1 and so forth. Due to this distribution policy, RMIDs may be visualized as localized to SNC domains, and there maybe cases where bandwidth is not counted. Consider for instance the case where thread with RMID 0 accesses will generate counts only for traffic in SNC domain 0. Any traffic from this thread that accesses other SNC domains will not increment any of the other counters. In other words, each SNC domain will get an equal number of distinct RMIDS from the global pool of RMIDS that are not shared.

Figure 7-4. Default Mode Demonstrating SNC-4 and RMID Distribution

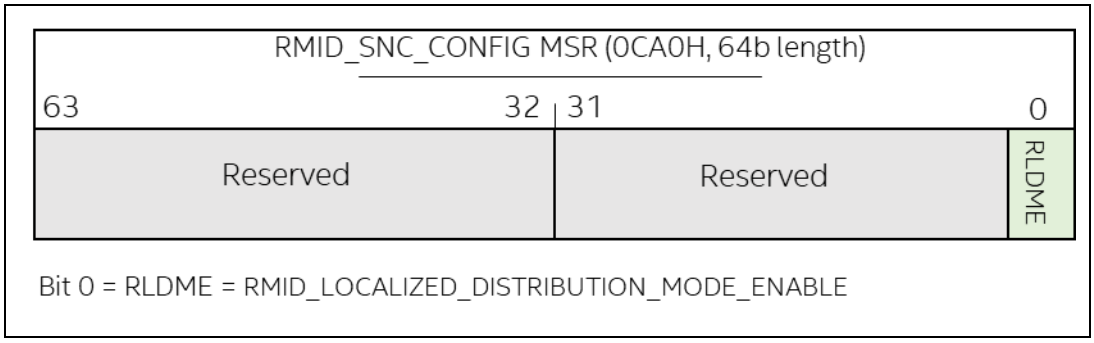


B.1.2.3.2 RMID Sharing Mode

RMID sharing mode allows the same RMID to be distributed with traffic accessing any and all SNC domains, but at the cost of a reduced number of SoC-level RMIDs available. This model-specific mode aims to mitigate the disadvantage of the Default mode where software should be aware of the RMID distribution per SNC domain (and NUMA-aware) and where traffic tagged with an RMID in one domain will not be counted if it accesses resources in another SNC domain. RMID sharing mode allows same RMID to sample across SNC domains, thus ensuring a complete count.

- This is an opt-in mode and requires that the software clears an enable bit defined in the following MSR 0XCA0, bit[0], see Figure 7-5. Note that as a model-specific capability, this mode is not guaranteed to be supported on all processors (see Appendix A.3 for support details).

Figure 7-5. The RMID_SNC_CONFIG MSR for Enabling RMID Sharing Mode



In this mode the number of RMIDs are distributed across all the L3 slices effectively reducing the number of RMIDs by the number of SNC domains. In the case of four SNC domains, the number of RMIDs are divided by four.

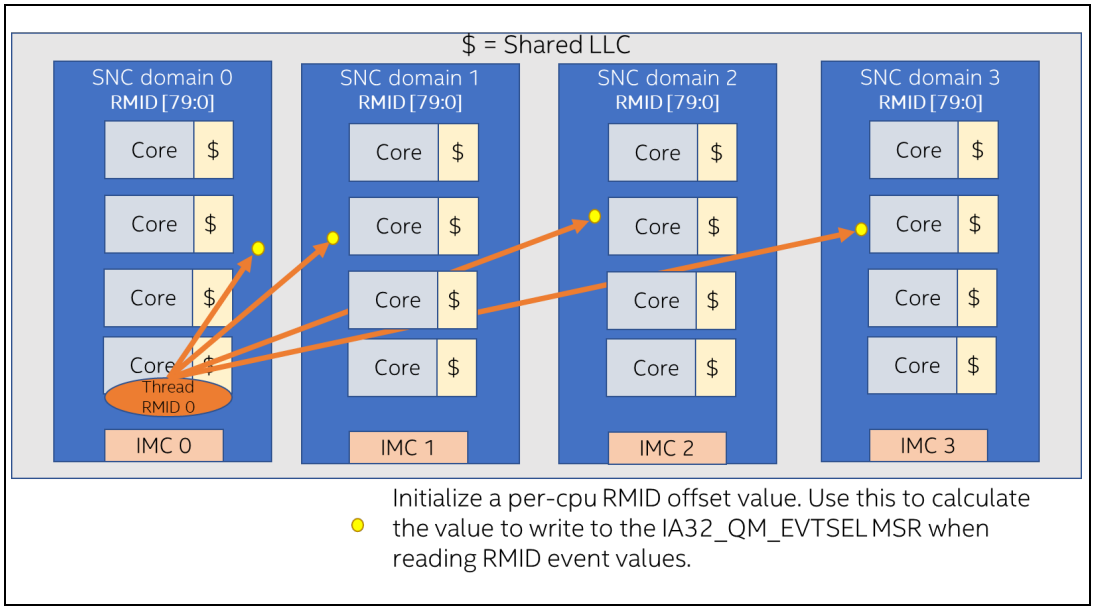
Number of valid RMIDs = (Highest RMID value/#SNC_clusters)

Using the previous example of 320 RMIDs, in this mode with SNC-2 enabled there would be (320/2), that is, 160 RMIDs, with SNC-4 enabled there would be (320/4), that is, 80 RMIDs.

Note: In SNC4 mode, to determine the count for RMID0, the count for RMID0, RMID80, RMID160, and RMID240 should be read and added to provide the total count for RMID0.

Note: It is the responsibility of software to read the values from each of the counters and calculate and interpret the sum using the output of the IA32_QM_CTR MSR. This is illustrated in Figure 7-6.

Figure 7-6. RMID Sharing Mode Demonstrating SNC-4 and RMID Distribution



B.1.2.4 Intel® RDT Software Considerations

Depending on its preferred use model and whether this model-specific capability is supported on a particular processor, software may select either the mode in which RMIDs are distributed or shared across the SoC and SNC domains. The default mode where each SNC cluster has a defined group of RMIDs or the opt-in mode which shares the same RMID across the SNC domains.

- Without SNC mode enabled the Remote Memory Bandwidth can be calculated by:
  - Remote Memory BW = (Total Memory BW – Local Memory BW) * Scaling Factor.
- With SNC Mode enabled software should scale the measured BW depending on the SNC_RMID Mode.
- CMT is similarly affected.

Table 7-2. SNC Enabled and RMID Distribution Mode Summary

	Default Mode	Opt-In : RMID Sharing Mode
Key highlights	• RMID_SNC_CONFIG MSR is Set. • Each SNC domain has its own group of RMIDs.	• RMID_SNC_CONFIG MSR is Clear. • Number of RMIDs divided by the number of SNC Domains. • Opt-In mode is enabled by software setting the MSR 0xCA0[0] = 0.
Example: **RMID Distribution per SNC Example for each Mode: SNC-4 config and Max 320 RMIDs**	1. For each SNC domain, the software should select an RMID from the range mentioned next to program IA32_PQR_ASSOC MSR. This range will be dependent on NUMA cluster you choose: • SNC_Domain_0 : RMID[79:0] • SNC_Domain_1 : RMID[159: 80] • SNC_Domain_2 : RMID[239:160] • SNC_Domain_3 : RMID[319:240] 2. To obtain monitoring data read via IA32_QM_EVTSEL, MSR uses only the RMID value to read counter value.	1. Number of Valid RMIDs = (#RMIDS/#SNC_Domains). Choose d in {0...79} in this example. **This range is used to program RMID field in the IA32_PQR_ASSOC MSR so that the appropriate hardware counters within the SNC domain are updated. 2. To obtain monitoring data via IA32_QM_EVTSEL MSR read 4 counter value from using the next formula: MAX_VALID_RMID = #RMIDS/#SNC_DOMAINS SNC_DOMAIN_0: RMID[0+d] SNC_DOMAIN_1: RMID[MAX_VALID_RMID*1 + d] SNC_DOMAIN_2: RMID[MAX_VALID_RMID*2 + d] SNC_DOMAIN_3: RMID[MAX_VALID_RMID*3 + d] For this example: SNC_DOMAIN_0: RMID[0+d] SNC_DOMAIN_1: RMID[80+d] SNC_DOMAIN_2: RMID[160+d] SNC_DOMAIN_3: RMID[240+d]
Differences	• Same number of RMIDS across SoC.	• RMIDS divided down by the number of SNC Domains and hence reduced number of RMIDS available for use.
Differences	• Miss traffic count due to software that traverses SNC domains. This can lead to inaccurate counts for CMT/MBM.	• Counts traffic that traverses SNC domains.
Differences	• Software needs to know the distribution of RMIDS to SNC domains.	• Software required to read all the RMID counters in the SNC domains and add up the individual count to get the final count.

Note: Only the monitoring features of Intel RDT are affected by the SNC feature. The allocation features, that is, CAT and MBA are not affected. Bit masks and BW targets apply globally across all domains. See Table 7-2 for SNC enabled and RMID distribution summary.

B.1.2.5 Scaling Factor Adjustment

CPUID-provided scaling factor (CPUID(0xF(Shared Resource Monitoring Enumeration leaf).0x1).EBX[31:0]), which software will use to convert MBM counts into bandwidth figures, needs adjustment in software when the system is configured in SNC mode. Moreover, calculating different types of bandwidths, such as local, total, or remote, also needs special considerations. This section describes how software needs to handle these special cases.

When using scaling factor under SNC mode, the scaling factor provided by CPUID will not account for the reduced number of L3 slices that will be handling local traffic. The scaling factor value will remain the same as any other clustering mode. software will then need to adjust the scaling factor. For this purpose, we define:

AdjustedScalingFactor = ScalingFactor / SNCClusterCount

B.1.2.6 SNC and Intel® RDT for Non-CPU Agent Implications

Intel RDT for non-CPU agents is affected similarly to traditional Intel RDT features in the presence of SNC. To obtain a correct CMT or MBM data sampling software should either localize I/O device memory allocations to a given cluster or sum RMID counts periodically, depending on the RMID localization mode selected.

In cases where multiple contexts are present on a device (SR-IOV, SIOV, with attached VMs that may span multiple SNC domains for their execution or multiple devices are behind an IOSF channel, if memory accesses are distributed across SNC clusters, then monitoring accuracy decreases considerably, and the risk of missing cache occupancy or memory bandwidth increases considerably.

SNC also affects I/O traffic. Software seeking to monitor I/O capacity or overflow BW to memory (I/O equivalent of CMT or MBM), should determine which SNC cluster a given address falls into using NUMA-aware supporting constructs (for example, ACPI HMAT, SLIT tables [4]) and pick a corresponding RMID for that cluster. As an example, if a device DMA write assigned to an RMID which does not land in the same SNC cluster as the address and its memory controller will not be tracked.

B.1.2.7 Calculating Local MBM Bandwidth per Cluster

When MSR 0xCA0 is set to 1 (Default Mode) software will be able to monitor local BW only from one SNC cluster. If MSR 0xCA0 is set to 0 (RMID Sharing Mode) then software will be able to monitor Local BW from all SNC clusters. Independent of the value in MSR 0xCA0, Local MBM Counts from a given SNC cluster can be converted to BW figures using the adjusted scaling factor following the same mechanism used under non-SNC modes:

*LocalMbmBwClusterN = (LocalMbmCountDeltaClusterN * AdjustedScalingFactor) / SampleTime*

Where:

- ‘LocalMbmCountDeltaClusterN’ = (Second Sample of LocalMbmCounter value (ClusterN) – First sample of LocalMbmCounter value (ClusterN).
- ‘AdjustedScalingFactor’ = ScalingFactor / SNCClusterCount.

B.1.2.8 Calculating Local MBM Bandwidth for Entire Socket

While operating under any non-SNC mode Local MBM BW will correspond to all the total traffic within the full socket. To obtain the same metric under SNC

mode software may add up the Local BW from each cluster. This can be achieved only when MSR 0xCA0 is set to 0. Otherwise, software will only be able to capture the local BW from a single cluster.

*LocalMbmBwSocket = ((LocalMbmCountDeltaCluster0 + ... LocalMbmCountDeltaClusterN) * AdjustedScalingFactor) / SampleTime*

Where:

- 'LocalMbmCountDeltaCluster0' = (Second Sample of LocalMbmCounter value (Cluster0) – First Sample of LocalMbmCounter value (Cluster0)… Similarly, delta for for each 1,2,…N.
- 'AdjustedScalingFactor' = ScalingFactor / SNCClusterCount.

B.1.2.9 Calculating Total MBM Bandwidth for the Socket

Calculating the Total MBM BW for the full socket, including the traffic from all clusters, will require that MSR 0xCA0 is set to 0.

*TotalMbmBwSocket = ((TotalMbmCountDeltaCluster0 + ... TotalMbmCountDeltaClusterN) * AdjustedScalingFactor) / SampleTime*

Where:

- 'TotalMbmCountDeltaCluster0' = (Second Sample of TotalMbmCounter value (Cluster0) – First Sample of TotalMbmCounter value (Cluster0)… Similarly, delta for each 1,2,…N.
- 'AdjustedScalingFactor' = ScalingFactor / SNCClusterCount.

B.1.2.10 Estimating Remote Traffic

As with Non-SNC modes, remote traffic can be estimated out of the socket's Total MBM BW and Local MBM BW with this simple relation:

RemoteMbmBwSocket = TotalMbmBwSocket – LocalMbmBwSocket

Calculating both TotalMbmBwSocket and LocalMbmBwSocket will require MSR 0xCA0 to be set to 0. However, if software decides to keep MSR 0xCA0 set to "1", its default value, an alternative mechanism exists to calculate the socket's MBM Remove BW as described in the following section.

B.1.2.11 Estimating Remote Bandwidth with MSR 0xCA0 set to 1

If software decides not to switch MSR 0xCA0 to value 0 (for example, out of Default mode) the mechanism described earlier to calculate the socket remote traffic will not work. However, it is still possible to estimate the remote traffic of the entire socket by using MBM counts from a single cluster.

*RemoteMbmBwSocket = (TotalMbmBwClusterN - LocalMbmBwClusterN) * SNCClusterCount*

B.1.2.12 Example for Local and Total MBM Bandwidth

In this example, software runs on a system configured in SNC-4 mode where CPUID(0xF(Shared Resource Monitoring Enumeration leaf).0x1).EBX[31:0]

reads 0x1E000 (ScalingFactor). AdjustedScalingFactor is then calculated to be 0x7800. If the system is configured with MSR 0xCA0=0 (RMID Distribution Mode) then software will have the ability to sample BW across all four clusters in this example. After sampling MBM counts with a delay of one second the following MBM Count increments are observed:

Table 7-3. Local and Total Count Increment

Cluster	Local MBM Count Increment	Total MBM Count Increment
0	174762	192238
1	43690	61166
2	0	17476
3	0	17476

Software can then calculate Local Bandwidth (BW), Total Bandwidth(BW) and Remote Bandwidth(BW) following these steps.

1. Calculate Local BW for Cluster 0 using the formula for LocalMbmBw described earlier:
   LocalMbmBwCluster0 = (LocalMbmCountDeltaCluster0 * AdjustedScalingFactor) / SampleTime
   LocalMbmBwCluster0 = (174762 * 0x7800) / 1
   LocalMbmBwCluster0 = 5368688640 B/s ~= 5GB/s
2. Calculate Total BW for Cluster 0 using the formula for TotalMbmBw described earlier:
   TotalMbmBwCluster0 = (TotalMbmCountDeltaCluster0 * AdjustedScalingFactor) / SampleTime
   TotalMbmBwCluster0 = (192238* 0x7800) / 1
   TotalMbmBwCluster0 = 5905551360B/s ~= 5.5GB/s
3. Following the same procedure Local and Total BWs for the different clusters may be calculated as shown in Table 7-4.

Table 7-4. Local and Total Bandwidth Example

Cluster	LocalMbmBwClusterN	TotalMbmBwClusterN
0	5 GB/s	5.5 GB/s
1	1.25 GB/s	1.75 GB/s
2	0	0.5 GB/s
	0	0.5 GB/s

4. We can also calculate the socket Local and Total BWs:
   LocalMbmBwSocket =((LocalMbmCountDeltaCluster0 + ... LocalMbmCountDeltaClusterN) * AdjustedScalingFactor) / SampleTime
   LocalMbmBwSocket = ((174762 + 43690 + 0 + 0) * 0x7800) / 1
   LocalMbmBwSocket = 6710845440B/s ~= 6.25GB/s
   TotalMbmBwSocket =((TotalMbmCountDeltaCluster0 + ... TotalMbmCountDeltaClusterN) * AdjustedScalingFactor) / SampleTime

TotalMbmBwSocket = ((192238 + 61166 + 17476 + 17476) * 0x7800) / 1
TotalMbmBwSocket = 8858296320B/s ~= 8.25GB/s

5. Finally, the remote BW for the socket can be estimated:
   RemoteMbmBwSocket = TotalMbmBwSocket – LocalMbmBwSocket
   RemoteMbmBwSocket = 8.25GB/s - 6.25GB/s ~= 2GB/s

We can also use this example to show to estimate the socket's remote BW if MSR 0xCA0 is set to 1 (Default mode). Under such conditions only MBM counts from a single cluster can be obtained. Assuming that the software has picked and RMID from cluster 0, we can use the values calculated earlier for LocalMbmBwCluster0 and TotalMbmBwCluster0. Then:

RemoteMbmBwSocket = (TotalMbmBwCluster0-LocalMbmBwCluster0) * SNCClusterCount
RemoteMbmBwSocket = (5.5GB/s – 5.0GB/s) * 4 ~= 2GB/s

Note that the value for RemoteMbmBwSocket obtained through this mechanism matches that obtained by using the MBM counts from all clusters.

By analyzing the results from this example, we can conclude, from the thread or threads assigned to the selected RMID that:

- Thread(s) are generating 5 GB/s of traffic towards cluster 0.
- Thread(s) are generating 1.25 GB/s of traffic towards cluster 1.
- Thread(s) are not generating local traffic towards clusters 2 or 3.
- Thread(s) are generating 2 GB/s of traffic towards a remote socket.
- Each SNC cluster is handling 0.5 GB/s of that remote traffic.

B.1.3 STLB QoS

Translation Lookaside Buffer (TLB) misses can cause costly execution delays due to page walks. Considered from a capacity management perspective, STLB QoS behaves in a similar manner as Cache Allocation Technology (CAT) does on the data caches, by giving software the ability to provide hints to hardware that guide the placement of translations in the STLB. This control can provide fair sharing or improved isolation of TLB resources between applications organized by Classes of Service.

Note: This model-specific feature is intended for use primarily with specialized real-time operating systems that provide extensions to bound the number of tasks running on a core and thus sharing a TLB. Depending on the software environment, additional runtime restrictions and software optimizations may be needed to observe the potential performance benefits of STLB QoS. Contact your Intel representative for additional details.

Refer to Appendix A.3 for supported product details, which vary across generation and processor type.

B.1.3.1 Enumerating Support for STLB QoS

STLB QoS is model specific and support for it is enumerated through the IA32_CORE_CAPABILITIES MSR. To determine if the processor supports the IA32_CORE_CAPABILITIES MSR, software can check whether the CPUID Extended Feature flag at CPUID.0x7(Structured Extended Feature Flags).0:EDX[30] is set to '1'.

If CPUID.0x7(Structured Extended Feature Flags).0:EDX[30] is '1', then support for STLB QoS can be confirmed via the IA32_CORE_CAPABILITIES MSR as defined next.

Table 7-5. STLB QoS Enumeration in IA32_CORE_CAPABILITIES MSR

Name	Address	Scope	Bit	RW	Bit Name	Description
IA32_CORE_CAPABILITIES	CFh	Core	0	RO	STLB_QOS	When set to 1, processor supports STLB QoS

B.1.3.2 STLB QoS Register Interfaces

This section contains the register interfaces for configuring STLB QoS. Software should first read the STLB_QOS_INFO to determine the maximum number of classes of service and capacity bitmask length and may then proceed to partitioning the STLB using the STLB_QOS_MASK_n registers.

B.1.3.2.1 STLB_QOS_INFO

Software may discover the necessary information for configuring STLB QoS via the STLB_QOS_INFO MSR as defined next.

Table 7-6. STLB_QOS_INFO MSR Definition

Name	Address	Scope	Bit	RW	Bit Name	Description
STLB_QOS_INFO	1A8Fh	Core	5:0	RO	NCLOS	Number of CLOS supported for STLB resource using minus-1 notation.
			19:16	RO	4K_2M_CBM	Length of capacity bitmask for 4K and 2M pages using minus-1 notation.
			29:29	RO	STLB_FILL_ TRANSLATION _MSR_SUPPORTED	MSR interface to fill STLB translations supported.
			30:30	RO	4K_2M_ALIAS	Indicates that 4K/2M pages alias into the same structure.

B.1.3.2.2 STLB_QOS_MASK_N

STLB_QOS_MASK_n registers define the capacity bitmask to be applied when filling new translations into the STLB. The mask used will depend on the core's current Class of Service at the time of TLB miss, as configured via the IA32_PQR_ASSOC MSR (covered in Chapter 3.2 Intel RDT Allocation Common Framework). The STLB_QOS_MASK_n registers are dynamic and may be changed at runtime.

Software should limit the number of mask registers used to the number of supported STLB QoS CLOS. For example, if STLB_QOS_INFO[NCLOS] returns 0x7, then a total of eight classes of service are supported and valid STLB_QOS_MASK_n registers would be 1A90h – 1A97h as defined in Table 7-7. Attempts to use unsupported STLB QoS mask registers will generate #GP(0).

Table 7-7. STLB_QOS_MASK_N MSR Definition

Name	Address	Scope	Bit	RW	Bit Name	Description
STLB_QOS_MASK_n	1A90h - 1A9Fh	Core	7:0	RW	WAY_MASK	STLB QoS mask for CLOS n. The number of mask bits is enumerated in MSR STLB_QOS_INFO. '1 in bit indicates allocation to the way is allowed. '0 indicates allocation to the way i' not allowed.[1,2]

NOTES: 1. Mask values must be contiguous 1s.
2. Way mask only applies to 4K/2M STLB.

B.1.3.2.3 STLB_FILL_TRANSLATION

As a further specialized extension to STLB QoS, certain processors support a mechanism to manually populate entries in the STLB, rather than requiring that pages of interest be accessed by software as part of a TLB fill flow to populate the entries. Note that this capability is not guaranteed to be supported on all future processors which support STLB QoS.

If STLB_QOS_INFO[STLB_FILL_TRANSLATION_MSR_SUPPORTED] is '1', software may populate entries in the STLB directly by writing the logical address (LA) and Class of Service to use for the fill to STLB_FILL_TRANSLATION as defined next.

Table 7-8. STLB_FILL_TRANSLATION MSR Definition

Name	Address	Scope	Bit	RW	Bit Name	Description
STLB_FILL_ TRANSLATION	1A8Eh	Core	3:0	WO	CLOS	Class of service to use for the fill.
			10:10	WO	X	Set to 1 when LA is to an executable page.
			11:11	WO	RW	Set to 1 when LA is to a writeable page.
			63:12	WO	LA	Logical address to use for fill.

Note: The STLB_FILL_TRANSLATION MSR should not be used in the VMX load list as a #GP(0) will occur.

B.1.4 L3 Cache Allocation Technology

Certain Intel® Core™ and Intel Atom® processors with support for Intel® Time Coordinated Computing (Intel® TCC), and certain communications related

Intel® Xeon® processors implement a model specific, non-architectural version of L3 Cache Allocation Technology (L3 CAT). In model-specific implementations, parameters such as CBM bitmask length and number of supported CLOS are specified on a per-processor basis rather than in CPUID (see the following section).

The non-architectural implementations of L3 CAT behave similarly to the architectural implementation, however under certain circumstances the performance characteristics may vary. Intel recommends evaluating overall system performance with model-specific non-architectural L3 CAT to verify performance targets are met.

B.1.4.1 Processor Support List

The following table can be used to identify which processors support the model specific non-architectural implementation of L3 CAT. Registers for programming the capacity bitmask for a given CLOS follow the same location and definition of the IA32_L3_MASK_n MSR's as defined in the Intel® Software Developer's Manual.

Table 7-9. Processor support list

Processor	Brand String	# L3 Classes of Service (CLOS)	Capacity Bitmask Length (CBM)
Intel Atom® Processors	Intel Atom® x6427FE Processor	4	16
	Intel Atom® x6425RE Processor		16
	Intel Atom® x6414RE Processor		16
	Intel Atom® x6212RE Processor		16
	Intel Atom® x6200FE Processor		8
	Intel Atom® X6416RE Processor		16
	Intel Atom® X6214RE Processor		16
	Intel Atom® x7211E Processor	16	12
	Intel Atom® x7425E Processor		12
	Intel Atom® x7213E Processor		12
11 Gen Intel® Core™ Processors (UP3-Series)	Intel® Core™ i7-1185GRE Processor	4	12
	Intel® Core™ i5-1145GRE Processor		8
	Intel® Core™ i3-1115GRE Processor		12
Intel® Xeon® W Processors (TGL-H)	Intel® Xeon® W-11865MRE Processor	4	12
	Intel® Xeon® W-11865MLE Processor		12
	Intel® Xeon® W-11555MRE Processor		8
	Intel® Xeon® W-11555MLE Processor		8
	Intel® Xeon® W-11155MRE Processor		8
	Intel® Xeon® W-11155MLE Processor		8
12 Gen Intel® Core™ Processors (S-Series)	Intel® Core™ i9-12900E Processor	16	12
	Intel® Core™ i7-12700E Processor		10

Processor	Brand String	# L3 Classes of Service (CLOS)	Capacity Bitmask Length (CBM)
	Intel® Core™ i5-12500E Processor		12
	Intel® Core™ i3-12100E Processor		12
13 Gen Intel® Core™ Processors (P-Series)	Intel® Core™ i7-1365UE Processor	16	12
	Intel® Core™ i7-1365URE Processor		12
	Intel® Core™ i5-1345UE Processor		12
	Intel® Core™ i5-1345URE Processor		12
	Intel® Core™ i3-1335UE Processor		12
	Intel® Core™ i3-1315UE Processor		10
	Intel® Core™ i3-1315URE Processor		10
	Intel® Core™ i7-1370PE Processor		12
	Intel® Core™ i7-1370PRE Processor		12
	Intel® Core™ i5-1350PE Processor		8
	Intel® Core™ i5-1350PRE Processor		8
	Intel® Core™ i3-1340PE Processor		8
	Intel® Core™ i3-1320PE Processor		8
	Intel® Core™ i3-1320PRE Processor		8
	Intel® Core™ i7-13800HE Processor		12
	Intel® Core™ i7-13800HRE Processor		12
	Intel® Core™ i5-13600HE Processor		12
	Intel® Core™ i5-13600HRE Processor		12
	Intel® Core™ i3-13300HE Processor		8
	Intel® Core™ i3-13300HRE Processor		8
13 Gen Intel® Core™ Processors (S-Series)	Intel® Core™ i9-13900E Processor	16	12
	Intel® Core™ i9-13900TE Processor		12
	Intel® Core™ i7-13700E Processor		12
	Intel® Core™ i7-13700TE Processor		12
	Intel® Core™ i5-13500E Processor		12
	Intel® Core™ i5-13500TE Processor		12
	Intel® Core™ i5-13400E Processor		10
	Intel® Core™ i3-13100E Processor		12
	Intel® Core™ i3-13100TE Processor		12

NOTES: 1. L3 CDP is not supported on any Intel® Core™ or Intel® Atom™ processors that implement model specific L3 CAT.
2. Communications-oriented processors from the Intel® Xeon® E5 v3 Family also support a form of model-specific L3 CAT.

B.1.4.2 Register Definitions

This section identifies deltas in the register definitions for programming model specific L3 CAT. The deltas are derived against the architectural equivalent register as documented in the Intel® 64 Architecture Software Developer's Manual (SDM), Volume 4: Chapter Title: MSRS IN THE 6TH GENERATION, 7TH GENERATION, 8TH GENERATION, 9TH GENERATION, 10TH GENERATION, 11TH GENERATION, 12TH GENERATION, AND 13TH GENERATION INTEL® CORE™ PROCESSORS, INTEL® XEON® SCALABLE PROCESSOR FAMILY, 2ND, 3RD, AND 4TH GENERATION INTEL® XEON® SCALABLE PROCESSOR FAMILY, 8TH GENERATION INTEL® CORE™ I3 PROCESSORS, AND INTEL® XEON® E PROCESSORS.

The naming convention for model specific L3 CAT registers mirrors the architectural L3 CAT registers without the "IA32_" prefix, for example, PQR_ASSOC (model specific) versus IA32_PQR_ASSOC (architectural).

The following deltas are consistent across all platforms that support model specific L3 CAT:

- Resource Monitoring ID's (RMIDs) are not guaranteed to be supported unless indicated by CPUID.
- L3 CDP is not supported.

B.1.4.2.1 PQR_ASSOC

The PQR_ASSOC MSR closely follows the IA32_PQR_ASSOC definition with exception of RMID. Platforms that support model specific L3 CAT typically do not support RDT Monitoring, with the exception of the Intel® Xeon® E5 v3 Family, and software should carefully consult CPUID before assuming support for any RDT Monitoring features.

B.1.4.2.2 L3_QOS_MASK_n

The L3_QOS_MASK_N MSRs are identical in definition to the IA32_L3_QOS_MASK_N for architectural L3 CAT. For the number of mask registers supported and acceptable CBM bit vector lengths, refer to Table 7-9 for the processor support list.

B.1.4.3 Shareable Bit Mask

Processors with an integrated GPU may be configured, by default, to allow the GPU full access to the L3 cache in certain performance modes. This behavior remains consistent independent of the values written to the L3_QOS_MASK_n registers, as these mask registers do not affect the cache policy for transactions initiated from the GPU. Software should consider all L3 cache ways as shared with the GPU.

For processors that support Intel® Time Coordinated Computing (Intel® TCC), optimizations are available for those that require improved isolation in the L3 cache. Contact your Intel representative for additional details.

B.1.4.4 Software considerations

Software that discovers enumerated support for architectural L3 CAT using CPUID.(EAX=07H(Structured Extended Feature Flags), ECX=0) will not automatically work with the non-architectural implementation. This section will cover known nuances and recommendations for working with the model specific non-architectural L3 CAT.

Note: Processors that support both L2 CAT and L3 CAT may have a delta in the number of CLOS supported between the L2 and L3. Intel recommends limiting software to use no more classes of service than the lesser of the two values.

B.1.4.4.1 Linux* Resource Control Groups (/sys/fs/resctrl)

Intel enables support for Intel RDT features in the Linux* kernel via Resource Control (CONFIG_X86_CPU_RESCTRL). Resource control provides an OS interface for configuring and using Cache Allocation Technology (CAT), Cache Monitoring Technology (CMT), Memory Bandwidth Monitoring (MBM), and Memory Bandwidth Allocation (MBA).

Resource Control leverages CPUID to detect hardware support for the various Intel RDT sub-features. On processors that support model specific non-architectural L3 CAT, CPUID.(EAX=07H(Structured Extended Feature Flags), ECX=0) will not enumerate support and therefore Resource Control will not support L3 CAT. Configuring of the L3_MASK_n registers will not be possible through the *resctrl* interface and must be completed through direct MSR access.

One feature of Resource Control is being able to associate a Class of Service with a Process Identifier (PID), and having the kernel automatically update the CLOS on context switch. If using a CPU that supports model specific non-architectural L3 CAT and updating the class of service on context switch is desired, it is possible to achieve this if the platform also supports L2 CAT. Resource Control would be utilized to configure L2 CAT and create the appropriate PID to CLOS mapping, while the L3 masks would need to be configured out-of-band (for example, direct MSR programming).

B.1.4.4.2 Intel-cmt-cat Tool (Intel RDT Utility)

The Intel RDT software package intel-cmt-cat is a software library that supports the Allocation and Monitoring features of Intel RDT. It can work with or without kernel support for RDT, which makes intel-cmt-cat a useful tool when working with model specific non-architectural L3 CAT.

The latest versions of the RDT Utility also include specialized print functions as command line options, which can be used to more easily decode the mapping of I/O devices to I/O RDT Channels for instance.

Intel-cmt-cat provides a pqos utility which access to the Intel RDT features through a command line interface. pqos can be used to program the L3_MASK_n registers on platforms that support non-architecture L3 CAT. Use the '--iface=msr' parameter to force enumeration and programming to be completed through MSR interfaces and not the OS interfaces.

The Intel RDT Utility is available at Github*:

https://github.com/intel/intel-cmt-cat

§