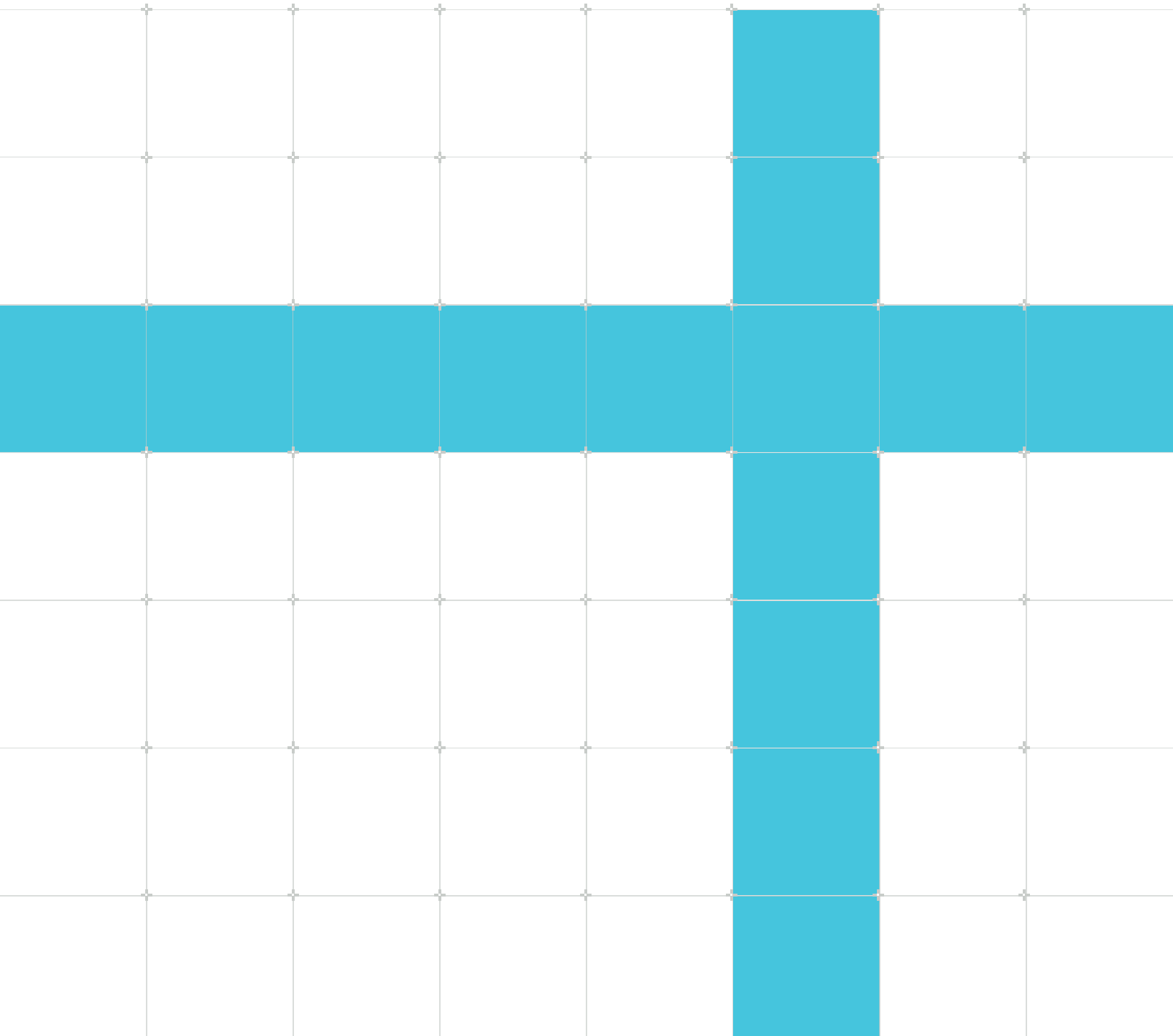
Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Hardware Guide

Version 1.0

Issue 01
109252_0100_01_en



Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Hardware Guide

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	12 July 2023	Non-Confidential	Initial release

Proprietary Notice

and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction **6**
1.1 Before you begin 6

2. MPAM information bundle transportation **7**
2.1 AMBA 5 Coherent Hub Interface architecture 7
2.1.1 AMBA 5 CHI Issue D and E 7
2.1.2 AMBA 5 CHI Issue F 8
2.1.3 Disabling MPAM from an MPAM enbaled Requester 9

3. Arm MPAM-enabled MSCs **10**
3.1 Arm DynamIQ Shared Unit-110 10
3.1.1 DSU-110 Overview 10
3.1.2 Cache-portion partitioning 10
3.1.3 Interaction with L3 cache power-down 12
3.1.4 Downstream propagation of the MPAM information bundle 15
3.2 Arm DynamIQ Shared Unit-120 15
3.2.1 DSU 120 Overview 15
3.2.2 Memory-bandwidth proportional-stride partitioning 16
3.3 Arm Neoverse CMN-700 Coherent Mesh Network 19
3.3.1 Overview 20
3.3.2 MPAM CBusy interaction 23
3.4 Arm CoreLink MMU-700 System Memory Management Unit 24
3.4.1 MMU-700 TCU Resource Instance selection control 25
3.5 Arm CoreLink GIC-700 Generic Interrupt Controller 26

4. MPAM System Guidance for Infrastructure **27**
4.1 Base System Architecture 27
4.2 General MPAM System Guidance 27
4.2.1 Number of PARTIDs to support 27
4.2.2 PARTID size 27
4.2.3 MPAM interrupts 28
4.3 Use of CBusy 28
4.4 Arm Neoverse Compute Subsystems 29

1. Introduction

This guide expands some of the Memory System Resource Partitioning and Monitoring (MPAM) concepts introduced in the Memory System Resource Partitioning and Monitoring Overview. The MPAM architecture defines several standard types of control and monitoring interfaces for memory-system resources. This guide will focus on:

- Control and monitoring interfaces implemented in Arm's MPAM-enabled IP.
- MPAM system-level design considerations

1.1 Before you begin

This guide assumes that you are familiar with Arm Exception levels, memory management, and the architectural description of Memory System Resource Partitioning and Monitoring. If you are unfamiliar with any of the previous topics, read the following guides before continuing with this guide:

- AArch64 exception model: Introduces the Exception and privilege model in AArch64.
- AArch64 memory management: Introduces the MMU, which controls virtual to physical address translation.
- Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture: Arm Architecture Reference Manual Supplement
- Memory System Resource Partitioning and Monitoring Overview: Introduces the MPAM architecture.

2. MPAM information bundle transportation

An MPAM-enabled system requires the transportation of an MPAM information bundle from an MPAM enabled Requester to Memory System Components (MSCs) around the system. This section describes the MPAM support added from Issue D of the AMBA 5 Coherent Hub Interface (CHI) architecture.

AMBA 5 CHI Architecture Specification describes the AMBA 5 Coherent Hub Interface (CHI) architecture.

2.1 AMBA 5 Coherent Hub Interface architecture

The MPAM_Support property indicates whether an interface supports MPAM. When the MPAM_Support property is FALSE or not specified the interface is not MPAM-enabled and the MPAM field width is zero.

When the MPAM_Support property is MPAM_9_1 the interface must include the MPAM field on the REQ and SNP channels.

CHI uses a slightly different naming convention then the *Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture Arm Architecture Reference Manual Supplement*. The following list describes the mapping between the CHI field name and their respective MPAM architectural name:

- PartID is equivalent to PARTID
- PerfMonGroup is equivalent to PMG
- MPAMNS is equivalent to MPAM_NS
- MPAMSP is equivalent to MPAM_SP

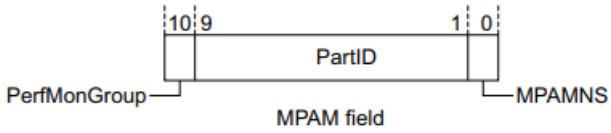
2.1.1 AMBA 5 CHI Issue D and E

AMBA 5 CHI Issues D and E support a 11-bit MPAM field. The following table shows how the 11-bit MPAM field is divided:

Table 2-1: AMBA 5 CHI.E 11-bit MPAM field

CHI MPAM field	Size
PartID	9 bits
PerfMonGroup	1-bit
MPAMNS	1-bit

Figure 2-1: AMBA 5 CHI.E 11-bit MPAM field



2.1.2 AMBA 5 CHI Issue F

To support the Realm Management Extension (FEAT_RME), the MPAM_NS component of the MPAM information bundle is redefined to be a 2-bit value, referred to as MPAMSP in CHI. To accommodate this change, AMBA 5 CHI Issue F supports a 12-bit MPAM field. The following table shows how the 12-bit MPAM field is divided:

Table 2-2: AMBA 5 CHI.F 12-bit MPAM field

CHI MPAM field	Size
PartID	9 bits
PerfMonGroup	1-bit
MPAMSP	2 bits

Figure 2-2: AMBA 5 CHI.F 12-bit MPAM field


Note

MPAMSP replaces MPAMNS from CHI Issue F onwards.

2.1.3 Disabling MPAM from an MPAM enbaled Requester

If an MPAM enabled Requester does not want to use MPAM, or if MPAM is disabled (MPAMn_ELn.MPAMEN == 0), the MPAM field must be set to a default setting. The following table shows how the default CHI MPAM fields are set:

Table 2-3: Default MPAM field

CHI MPAM field	Default value
PartID	0x0
PerfMonGroup	0x0
MPAMNS	Same as NS value in the Snoop / Request message
MPAMSP	Same as PAS value in the Snoop / Request message

3. Arm MPAM-enabled MSCs

This chapter describes several Arm MSCs and their MPAM functionality.

3.1 Arm DynamIQ Shared Unit-110

Arm DynamIQ Shared Unit-110 (DSU-110) compatible Arm processor cores, such as the Cortex-X2 include support for FEAT_MPAM, and as such are MPAM-enabled Requesters.

The Arm DSU-100 and compatible cores do not support FEAT_MPAM.

3.1.1 DSU-110 Overview

The DSU-110 L3 cache is an MPAM-enabled MSC, which implements a cache-portion partitioning control interface.

The following table shows the DSU-110 MPAM information bundle properties:

Table 3-1: DSU-110 MPAM information bundle properties

MPAM information bundle	Size
PARTID	6 bits
PMG	1-bit
MPAM_NS	1-bit

- The DSU-110 does not support FEAT_RME and implements only two PARTID spaces: Non-secure PARTID space and Secure PARTID space. The 1-bit MPAM_NS communicates this.
- PMG is not used internally by DSU, but is transported downstream of DSU to the external memory system.

3.1.2 Cache-portion partitioning

A portion of a memory system resource is a uniquely identifiable part of the resource. A particular resource has a constant number of portions and all portions of a resource are the same size. For example, a cache has a defined number of ways.

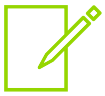
For the DSU-110, the number of L3 cache ways depends on the size of the L3 cache:

- When selecting a power-of-two L3 cache size of 256KB, 512KB, 1024KB, 2MB, 4MB, 8MB, or 16MB each cache slice has 16 ways.
- When selecting a non-power-of-two L3 cache size of 1536KB, 3MB, 6MB, or 12MB, each cache slice only has 12 ways.

The partitioning of the DSU-110 L3 cache is done by groups of cache ways. For the DSU-110 each group contains two ways, so a maximum of 8 partitions are supported. When programming the partitioning, the groups of L3 cache way pairs are referred to as portions.

The following table indicates the mappings between portions and L3 cache ways:

Portion	Ways
0	0, 1
1	2, 3
2	4, 5
3	6, 7
4	8, 9
5	10, 11
6	12, 13
7	14, 15


Note

Only power-of-two L3 cache sizes support 16 ways. Non-power-of-two L3 cache sizes support only 12 ways.

You can attribute portions of a cache to a PARTID partition value. Only cache line requests with that PARTID value to allocate into those portions of the cache. The partitioning only affects the cache line allocation. If the cache line is already allocated in the cache in a different portion, then a cache hit still occurs.

When a request to a cache requires a cache line to be allocated in the cache, the PARTID of that request identifies the portions of the cache where the request could allocate the line.

The DSU-110 implements a cache-portion bitmap (CPBM) control interface. A CPBM controls the cache-storage portion allocation for a partition.

In a CPBM, the control setting is a bitmap in which each bit corresponds to a particular portion of the resource. Each bit grants the PARTID using this control setting to allocate the portion corresponding to that bit.

You can allocate portions for the exclusive use of a partition or shared between two or more partitions.

Consider the following allocation of cache portion to PARTID:

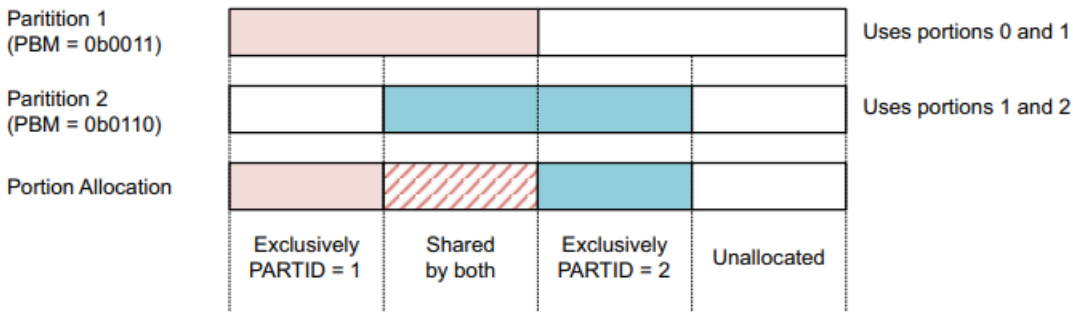
PARTID	Portions
1	0, 1
2	1, 2

The above allocation will mean:

- Portion 0 is exclusive to allocations made by PARTID `0x1`.
- Portion 1 is shared between both PARTIDs.
- Portion 2 is exclusive to allocations made by PARTID `0x2`.

The following diagram shows how the cache potions are allocated to a PARTID.

Figure 3-1: CPBM shared and exclusive allocations



Note

If a transaction is mapped to a partition for which the MPAMCFG_CPBM setting has no portions set, then this transaction is not allocated into the L3 cache.

3.1.3 Interaction with L3 cache power-down

You can independently power down parts of the L3 cache RAM to reduce RAM leakage power when not in use.

The L3 cache RAM power-down feature allows the RAMs to be powered down in groups of ways, giving options of 100%, 50%, or 0% of the L3 cache capacity.

This reduction in cache ways can degrade the performance when there are insufficient ways available to a process. For example, if you only have 50% of the L3 cache capacity, then the number of ways are halved in each L3 cache partition portion.

Note

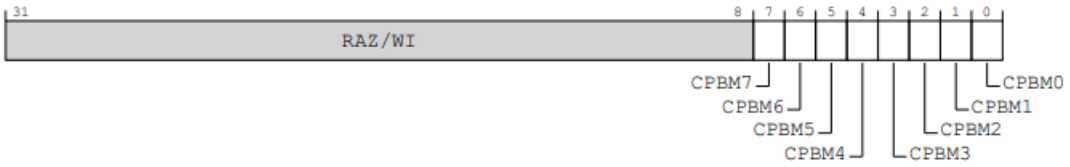
Arm recommends that you take care when powering down cache ways while using cache partitioning. Powering down cache ways will reduce the number of portions available to a partition.

3.1.3.1 Security considerations

The Secure and Non-secure states have separate control registers for programming the cache portions (cache ways) that are assigned to each partition ID (PARTID).

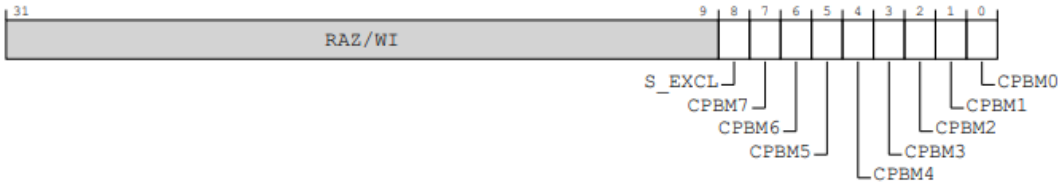
The MPAMCFG_CPBM register programs the cache portions that can be used by each of the different Non-secure state partition ID values.

Figure 3-2: MPAMCFG_CPBM encoding



The MPAMCFG_CPBM_s register programs the cache portions that can be used by the different Secure state partition ID values. The Secure state partition control register, MPAMCFG_CPBM_s, has an additional non-architectural control bit, MPAMCFG_CPBM_s.S_EXCL. This bit enabled the Secure state partitioning programming to override the Non-secure state partitioning programming.

Figure 3-3: MPAMCFG_CPBM_s encoding



Note

When MPAMCFG_CPBM_s.S_EXCL is set to 1, any of the cache portions used for a Secure partition ID are only permitted to allocate transactions from the Secure state. If any of the Non-secure state partition IDs have been programmed to use these cache portions, then Non-secure state transactions are not permitted to allocate into these L3 cache portions.

3.1.3.2 Example configuration

In the following example, the EL3 PARTID Space Manager configures the MPAM information bundle as follows:

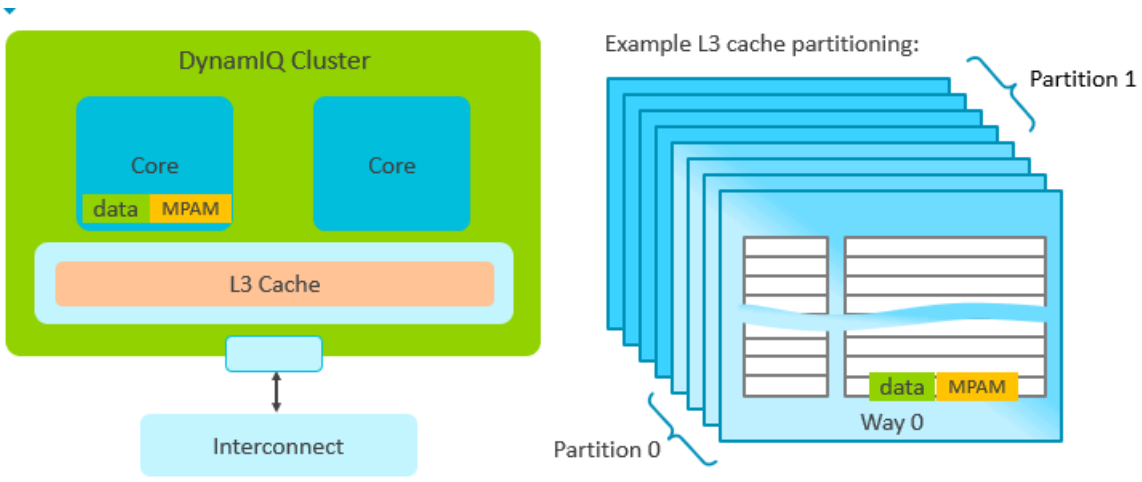
```
// Enable MPAM at EL3
// ---------------
MRS     x0, MPAM3_EL3
BIC     x0, x0, #(0x1<<60)        // EL3 FORCE_NS = 0x0
ORR     x0, x0, #(0x2<<16)        // EL3 PARTID_D == 0x2
ORR     x0, x0, #(0x2<<0)         // EL3 PARTID_I == 0x2
ORR     x0, x0, #(1<<63)          // Set MPAMEN == 0x1
MSR     MPAM3_EL3, x0
ISB
```

With the above configuration, while executing at EL3, any data memory accesses or instruction fetches are sent with the below configured MPAM information bundle:

MPAM information bundle	Value
PARTID_D	0x2
PARTID_I	0x2
MPAM_NS	0x0 (Secure)

The DSU-110 implements a CPBM control interface that can be configured to give a portion of the L3 cache to a specific partition ID space. In the following example, EL3 data memory access and instruction fetches are restricted to allocation in partition 0 of the L3 cache (ways 0, 1, 2 and 3):

Figure 3-4: DSU-120 Cache-portion bitmap (CPBM) control interface example




```
LDR     x1, =DSU_MPAMCFG_PART_SEL_s
MOV     w0, #2                    // Set MPAMCFG_PART_SEL_s.PARTID_SEL = 0x2
STR     w0, [x1]                  // Write MPAMCFG_PART_SEL_s
DSB     SY
LDR     x1, =DSU_MPAMCFG_CPBM_s
MOV     w0, #3                    // Set MPAMCFG_CPBM_s.CPBM0 (ways 0, 1, 2 and 3)
```

```
STR    w0, [x1]                 // Write MPAMCFG_CPBM_s
DSB    SY
```

3.1.4 Downstream propagation of the MPAM information bundle

The MPAM architecture requires that when a cache line is evicted to a downstream cache, the evicting cache must produce the MPAM information that is associated with the cache line.

A System-Level ache (SLC) is permitted to propagate either:

- the MPAM information of the request that caused the eviction
- the stored MPAM information associated with the evicted line

The DSU-110 includes the L3_MPAM_STORAGE parameter. When L3_MPAM_STORAGE = TRUE, the L3 cache stores the MPAM information bundle, which is retrieved on evictions.

Storing the MPAM information in the L3 cache is recommended if there is a downstream cache which supports MPAM.

If the system only requires valid MPAM information values for read transactions, or the system includes no downstream SLC, then this MPAM information bundle storage is not required.


Note

If the MPAM information is not being stored, then any L3 evictions use the MPAM information of the transaction that causes the eviction.

3.2 Arm DynamIQ Shared Unit-120

The Arm DynamIQ Shared Unit-120 (DSU-120) expands on the MPAM support provided with the DSU-110. As with the DSU-110, the DSU-120 supports a cache-portion partitioning control interface. Also, the DSU-120 apportions bandwidth differently to different sources (PEs or ACP Requesters), based on the Memory System Resource Partitioning and Monitoring (MPAM) bandwidth partitioning control interface. This interface is architecturally referred to as a Memory-bandwidth proportional-stride partitioning control interface.

3.2.1 DSU 120 Overview

Bandwidth partitioning enables you to control how bandwidth is split when the demand for bandwidth is greater than the bandwidth available.

By default, the bandwidth available within the DSU-120 should be distributed fairly between all cores and ACP requesters making external memory requests. However, there might be circumstances when more control is required. For example, in a dual-core cluster with two Accelerator Coherency Port (ACP) interfaces, each core and each ACP interface would get one

quarter of the bandwidth. But, allowing both ACP interfaces collectively to use up half of the overall bandwidth, might impact on the performance of the cores. Therefore, the ACP can be restricted to using only a smaller proportion of the overall bandwidth.

The size of the DSU-120 MPAM information bundle depends on whether RME is enabled. For RME to be enabled, the cluster must be in Direct connect configuration and the LEGACYTZEN input signal is LOW.

3.2.1.1 RME disabled

The following table shows the DSU-120 MPAM information bundle properties when the DSU-120 is not in Direct connect mode and the input signal LEGACYTZEN is LOW:

MPAM information bundle	Size
PARTID	6 bits
PMG	1-bit
MPAM_NS	1-bit

3.2.1.2 RME enabled

The following table shows the DSU-120 MPAM information bundle properties when the DSU-120 is in Direct connect mode, RME is supported by the core, and the input signal LEGACYTZEN is LOW:

MPAM information bundle	Size
PARTID	6 bits
PMG	1-bit
MPAM_SP	2 bit


Note

When DSU-110 or DSU-120 is in Direct connect mode, no L3 cache is present and no MPAM partitioning interface is implemented.

3.2.2 Memory-bandwidth proportional-stride partitioning

Each MPAM PARTID has a separate MPAMCFG_MBW_PROP register, which contains an enable bit and the STRIDEM1 field. If the bandwidth partitioning is enabled for that MPAM PARTID, the 6-

bit STRIDEM1 value controls how much bandwidth to give to ACP transactions and cores that are using that PARTID.

The STRIDEM1 value is the reciprocal of the relative bandwidth required, minus one. The following python code shows how to calculate the STRIDEM1 values based on a ratio of relative bandwidth:

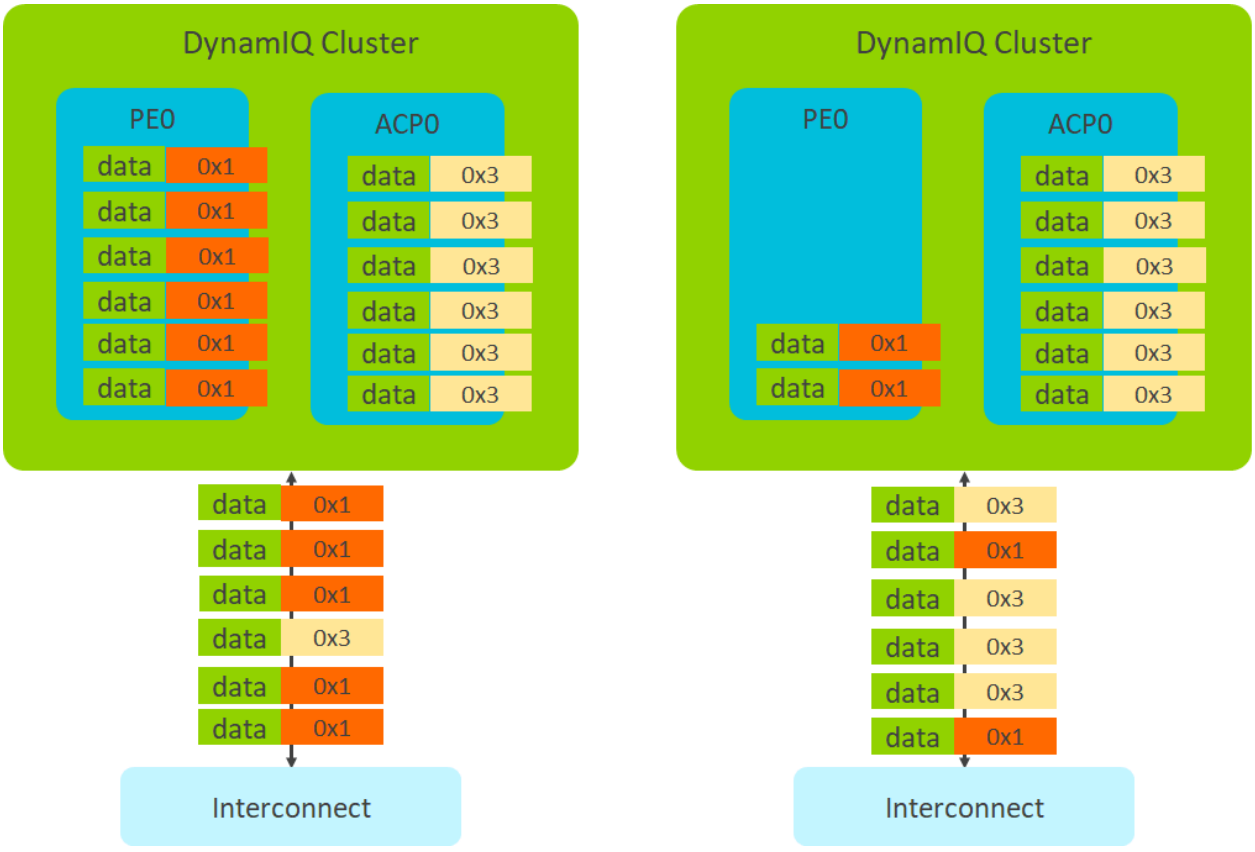
```
def stridem1(ratios: list):
    base = max(ratios)
    rel_bw = [r/base for r in ratios]
    return [1/rbw - 1 for rbw in rel_bw]
```

3.2.2.1 Example configuration

The DSU-120 implements a Memory-bandwidth proportional-stride partitioning control interface. The Memory-bandwidth proportional-stride partitioning control interface uses the MPAMCFG_MBW_PROP_s or MPAMCFG_MBW_PROP_ns registers to assign a relative bandwidth ratio to a partition, as programmed by the corresponding MPAMCFG_PART_SEL register.

The scheme only limits the bandwidth available to a PARTID when there is contention for that bandwidth. In the example shown below, ACP0 might be able to use more than the 1/6 allocated if PE0 was not generating sufficient traffic to consume the other 5/6 of the bandwidth.

Figure 3-5: DSU-120 Memory-bandwidth proportional-stride partitioning control example



It is very unlikely in a real system that PE0 and ACP0 would utilize sufficient bandwidth to trigger Memory-bandwidth proportional-stride partitioning. This is because the bandwidth partitioning mechanism is work-conserving, which means that enabling it does not reduce the total bandwidth that the cluster uses.

The scheme only regulates PARTIDs that are using more than their fair share of bandwidth. Therefore, if a PARTID is not attempting to use much bandwidth, then this does not reduce the ability of other PARTIDs to use that bandwidth.

A PARTID that is already getting all the bandwidth that it wants does not gain more bandwidth with a lower STRIDEM1 value.

If there is spare bandwidth, the bandwidth partitioning does not regulate the bandwidth of any PARTIDs.

Using the above calculation, 'stridem1' we can see the resultant STRIDEM1 values are as follows:

```
stridem1([10, 10, 2, 2]) = [0.0, 0.0, 4.0, 4.0]
```

The following table shows a static mapping between Requesters and PARTIDs, their given ration and the corresponding STRIDEM1 value that must be reflected in MPAMCFG_MBW_PROP_s.STRIDEM1 register.

Requester	PARTID	Ratio	STRIDEM1
PE0	0x1	10	0
PE1	0x2	10	0
ACP0	0x3	2	4
ACP1	0x4	2	4

Here is an example programming sequence to limit ACP0 (PARTID == 0x3) to 1/6 of the total bandwidth utilization.

```
LDR    x1, =DSU_MPAMCFG_PART_SEL_s
MOV    w0, # 0x3
STR    w0, [x1]                      // Write MPAMCFG_PART_SEL_s
DSB    SY
LDR    x1, =DSU_MPAMCFG_MBW_PROP_s
MOV    w0,  # 0x4                     // Set MPAMCFG_MBW_PROP_s.STRIDEM1
ORR    w0, w0, #(0x1 << 31)           // Set MPAMCFG_MBW_PROP_s.EN
STR    w0, [x1]                       // Write MPAMCFG_MBW_PROP_s
DSB    SY
```

3.3 Arm Neoverse CMN-700 Coherent Mesh Network

The CMN-700 supports Memory System Performance Resource Partitioning and Monitoring (MPAM). MPAM features enable software to optimize the use of memory resources and to monitor how those resources are used.

MPAM support is enabled when the build-time configuration parameter CHI_MPAM_ENABLE is set.

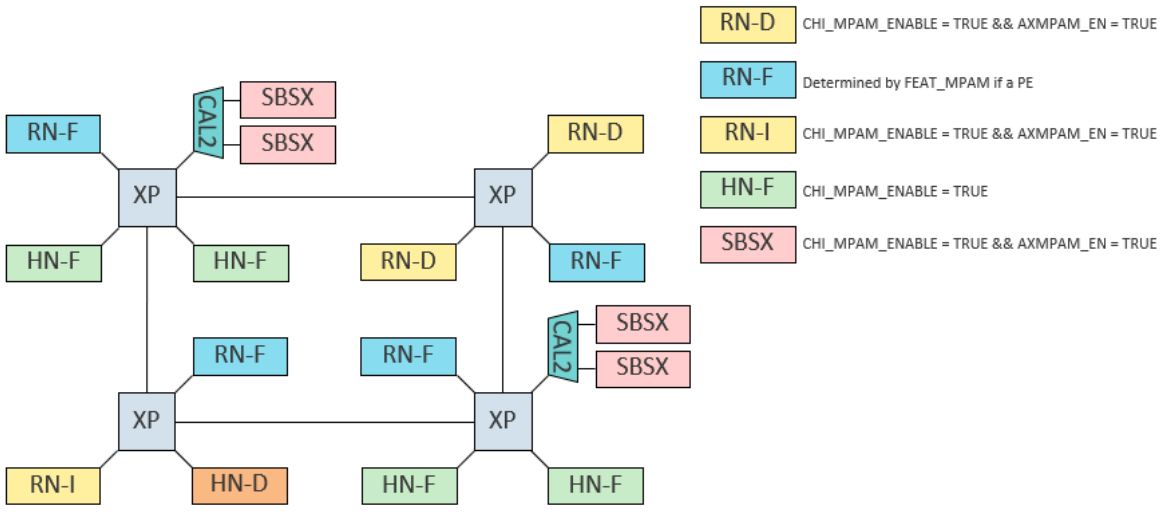
When enabled, CMN-700 XPs, RN-I/-D, HN-F, HN-I/-T/-D/-P, RA, HA, and SBSX nodes will propagate the MPAM information bundle.

Note

The AXMPAM_EN build-time configuration parameters determine MPAM support on ACE-Lite interfaces, per node.

When MPAM is enabled, CMN-700 propagates the MPAM information bundle throughout the network. Various nodes are designed to propagate the MPAM field when processing requests. Which nodes are able to propagate the MPAM information bundle depends on configuration parameters.

Figure 3-6: Configuration requirements for propagating the MPAM information bundle around the CMN-700



3.3.1 Overview

Unlike the DSU, properties of the MPAM information bundle are configurable. The following list describes the CMN-700 HN-F build-time configuration parameters for the MPAM information bundle :

- MPAM_NS_PARTID_MAX: number of Non-secure PARTID values supported (1-512, default 64)
- MPAM_S_PARTID_MAX: number of Secure PARTID values supported (1-512, default 16)
- MPAM_NS_PMG_MAX: number of Non-secure performance monitor groups supported (1 or 2, default 2)
- MPAM_S_PMG_MAX: number of Secure performance monitor groups supported (1 or 2, default 2)

The CMN-700 System-Level Cache (SLC) is an MPAM-enabled MSC. The SLC supports the following Memory-system resource partitioning interfaces:

- Cache-portion partitioning
- Cache maximum-capacity partitioning

The SLC supports the following Memory-system resource usage monitoring interfaces:

- Cache-storage usage monitor

The SLC is split into multiple SLC slices, with one slice per HN-F node. Each HN-F node has three configuration register blocks:

- HN-F configuration registers
- HN-F MPAM Secure configuration registers
- HN-F MPAM Non-Secure configuration registers

Each HN-F node in the mesh must have all three of its configuration register blocks programmed independently

MPAM capacity partitioning and portion partitioning are disabled by default, even when the CHI_MPAM_ENABLE parameter has been set. At boot time, the following configuration register bits should be set in each HN-F node to enable the MPAM partitioning features:

- cmn_hns_cfg_ctl.hnf_slc_mpam_ccap_enable set to `0b1` to enable
- cmn_hns_cfg_ctl.hnf_slc_mpam_cpor_enable set to `0b1` to enable

3.3.1.1 Cache-portion partitioning

For the CMN-700, the number of cache-portions is defined by the number of SLC cache ways. The number of SLC cache ways depends on the size of the SLC cache:

- When selecting a power-of-two SLC cache slice size of 128KB, 256KB, 1MB, 2MB or 4MB, each SLC slice has 16 ways.

- When selecting a non-power-of-two SLC cache slice size of, 384KB, 1.5MB or 3MB each cache SLC slice has 12 ways.

The CMN-700 supports address-based cache locking (lockdown), including On-Chip Memory (OCM). When address-based cache locking is used, locked ways are not available for cache-portion partitioning. That is, locked ways are not available as a cache-portion partition.

The CMN-700 SLC supports the Half-Associativity Mode (HAM) power state, where the SF is enabled but the upper half of the SLC ways are disabled and powered off. In HAM mode, cache capacity is adjusted for half the cache, where portions 15:8 are aliased to portions 7:0.


Note

Software can dynamically determine the number of available SLC portions by reading corresponding SLC slice MPAMF_CPOR_IDR.CPBM_WD.

The following example, the CMN-700 cache-portion partitioning interface is programmed to provide 2 portions (way 0 and way 1) of the SLC slice to partition (PARTID = `0x8`, MPAM_NS == `0x0`):

```
LDR    x1, =cmn_hns_s_mpamcfg_part_sel
MOV    w0, #0x3                      // Set cmn_hns_s_mpamcfg_part_sel.partid_sel =
 0x8
STR    w0, [x1]                      // Write cmn_hns_s_mpamcfg_part_sel
DSB    SY
LDR    x1, =cmn_hns_s_mpamcfg_cpbm
MOV    w0, #3                        // Set cmn_hns_s_mpamcfg_cpbm.cbpm = 0x3 (way 0
 and way 1)
STR    w0, [x1]                      // Write cmn_hns_s_mpamcfg_cpbm
DSB    SY
```

3.3.1.2 Cache maximum-capacity partitioning

Using Cache maximum-capacity partitioning you can set a limit on the storage capacity of a cache that a partition may use. Setting a maximum cache capacity to a partition enables requests attributed to that partition to allocate up to that maximum cache capacity. Attempts to allocate beyond the configured maximum capacity result in limiting the ability to allocate.

The CMN-700 limits a partition's ability to allocate by preventing further allocation into the SLC. Additional allocations can only occur if a victim cache line in the same partition (matching PARTID and MPAM_NS) is found within the set. Otherwise, if a matching victim is not found, clean lines are dropped and dirty lines will cause a write to the memory controller.

The cache maximum-capacity control setting is programmed by storing a capacity limit into the MSC's cache maximum capacity control interface, MPAMCFG_CMAX. The cache maximum-capacity limit is a fraction of the cache's total capacity. The format of the limit value is a fixed-point fraction. This setting is a hard limit on the cache capacity.

The CMN-700s SLC provides a CMAX granularity of 0.78% (1 / 2^7). Therefore, the CMAX granularity size depends on the size of the SLC slice, and can be calculated as follows:

Figure 3-7: CMAX granularity calculation

$$CMAXgranularity = \frac{SLCslicesizebytes}{128}$$

Example calculation for a 2MB SLC cache slice:

```
CMAX granularity = 2048 / 128
CMAX granularity = 16KB
```

Therefore, for a 2MB SLC cache slice, the cache maximum-capacity limit for a partition is defined by the MPAMCFG_CMAX.CMAX x 16KB.

The CMN-700 cache maximum-capacity control interface also provides a soft limit. The soft limit is a programmable percentage below CMAX (default is 3.13% below CMAX for a PARTID). If CMAX value set is at or below 12.5%, the soft limit is ignored. If a partition's occupation goes above the soft limit, its allocated cache lines become more likely to be evicted.

The following example, executed in EL3 shows how the CMN-700 cache maximum-capacity control interface is configured to provide 15.6% (20 * 0.78%) of an SLC slice (2MB) to a partition (PARTID = `0x3`, MPAM_NS == `0x0`):

```
LDR    x1, =cmn_hns_s_mpamcfg_part_sel
MOV    w0, #0x3                      // Set cmn_hns_s_mpamcfg_part_sel.partid_sel =
 0x3
STR    w0, [x1]                      // Write cmn_hns_s_mpamcfg_part_sel
DSB    SY
LDR    x1, =cmn_hns_s_mpamcfg_cmax
MOV    w0, #20
LSL    w0, w0, #9                    // Set cmn_hns_s_mpamcfg_cmax.cmax = 20
STR    w0, [x1]                      // Write cmn_hns_s_mpamcfg_cmax
DSB    SY
```

3.3.1.3 Cache-storage usage monitors

MPAM resource monitors provide software with measurements of the resource-type usage that can be partitioned by MPAM. Each type of monitor measures the usage by memory-system transactions of a PARTID and PMG.

The HN-F MPAM_NUM_CSUMON build-time parameter determines how many monitors are supported (1-16, default 4). The MSMON_CFG_MON_SEL register selects the monitor instance to access through the MSMON configuration and counter registers.

Each cache-storage usage monitor can count SLC cache capacity used by:

- Single PARTID or all PARTIDs
- Single PMG or both PMG

- Any combination of the above

This is programmed using the filter register, the MSMON_CFG_CSU_FLT that sets the PARTID and PMG to be monitored.

Cache-storage usage capacity can be read in real time, or captured and read at a later point in time:

- The cache-storage usage register MSMON_CSU reports the current amount of storage currently present within the cache allocated by the PARTID and PMG, as defined by MSMON_CFG_CSU_FLT.
- The cache-storage usage register MSMON_CSU_CAPTURE reports the captured amount of storage reported by the MSMON_CSU within the cache allocated by the PARTID and PMG, as defined by MSMON_CFG_CSU_FLT.

Storage capacity is reported in number of bytes.

The MSMON_CSU_CAPTURE register is updated by a capture event. The CMN-700s cache-storage usage monitors support two capture events.

- External capture event 6 - is triggered using PMUSNAPSHOT interface
- Local capture event 7 - generated by a software write to MSMON_CAPT_EVNT

Multiple capture events cannot be triggered within 32 cycles of each other.

3.3.2 MPAM CBusy interaction

The CMN-700 does not implement any of the bandwidth management features of MPAM. Any of the MPAM bandwidth features in the external memory system needs to be supported and performed by the SN-F nodes (memory controllers).

In CMN-700, the only place where CBusy and MPAM intersect is in CBusy reporting options by the HN-F based on SN-F CBusy values.

You can configured HN-F nodes to filter the SN-F CBusy value by MPAM PartitionID+MPAM_NS value. For example, a request with a REQ.MPAM=`0x3` will receive in its response the last SN-F CBusy value provided for a request with REQ.MPAM=`0x3`.

This can occur whether or not the incoming request actually required a memory controller access, because the HN-F nodes records the most recent SN-F CBusy value for each PartitionID supported.

3.4 Arm CoreLink MMU-700 System Memory Management Unit

The CoreLink MMU-700 System Memory Management Unit implements Memory System Resource Partitioning and Monitoring (MPAM).

The MMU-700 provides MPAM enablement to upstream Requesters that are otherwise be incapable of sending an MPAM information bundle downstream. The MPAM information bundle appended by the MMU-700 can be used to partition any downstream MPAM-enabled MSC, and also memory system resources within the MMU-700 itself.

The MMU-700 can also append an MPAM information bundle to transactions that it generates on its own (from the TCU) into the memory system, for example, page or Configuration Table Walk accesses, SMMU queue access, MSIs etc.

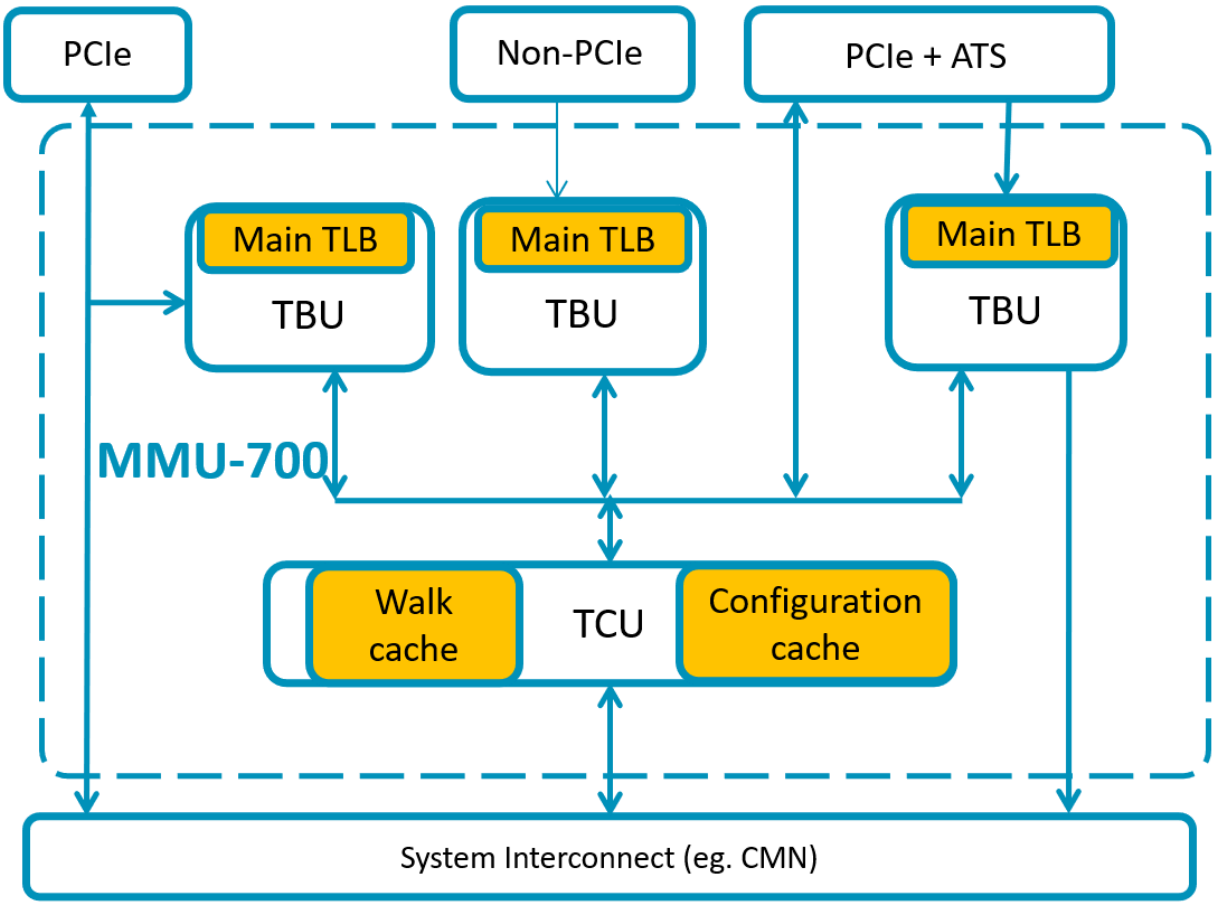
When translation is enabled by SMMU_(S_)CR0.SMMUEN == 1, the the STE and CD configure the MPAM information bundles of the given stream (StreamID/SubstreamID).

The Arm System Memory Management Unit Architecture Specification, SMMU architecture version 3 describes the assignment of an MPAM information bundle.

The MMU-700 supports capacity partitioning of the following MMU-700 resources.

- TBU MTLB
- TCU Configuration Cache Block (CCB)
- TCU Walk Cache Block (WCB)

Figure 3-8: Partitionable resources in the MMU-700



3.4.1 MMU-700 TCU Resource Instance selection control

The MMU-700 uses Resource Instance selection (RIS) to select which of the MMU-700 TCU resources are being configured: WCB or CCB.

RIS allows support for MSCs with multiple resources. This includes multiple resources with the same resource type or partitioning control. This means that each MSC can only have independent resource controls and two or more resources of the same type when RIS is implemented.

Each resource that has MPAM resource partitioning controls, or can be monitored by an MPAM resource usage monitor, has a RIS value.

The RIS value in MPAMCFG_PART_SEL.RIS selects which resource to describe in ID register fields.

MPAMCFG_PART_SEL.RIS and MPAMCFG_PART_SEL.PART_SEL to select the resource and PARTID when accessing MPAMCFG_* resource control registers.

For the MMU-700, the following table outlines the relationship between RIS values and partitionable resources within the MMU-700 TCU:

Partitionable resource	RIS value
Walk Cache Block	0
Configuration Cache Block	1

The MMU-700 TBU only contains one partitionable resource, the MTLB, so RIS is not required.

3.5 Arm CoreLink GIC-700 Generic Interrupt Controller

The Arm CoreLink GIC-700 Generic Interrupt Controller supports Memory Partitioning and Monitoring (MPAM). The GIC-700 can assign an MPAM information bundle to all memory accesses that it issues on the ACE5-Lite manager interface.

The GIC-700 ACE5-Lite manager interface issues read transactions from the:

- ITS, which accesses ITS command queue and ITS tables.
- Redistributors, which access the LPI configuration tables and LPI pending tables.

The GIC-700 GICR_PARTIDR is global for all Redistributors. It is used to configure the MPAM information bundle for each PEs Redistributor on a chip. Therefore, all of the Redistributors will generate the same MPAM information bundle.

The MPAM information bundle generated by read transactions from the ITS is configured independently for each ITS using the GITS_PARTIDR.

Both the GICR_PARTIDR and GITS_PARTIDR only allow software to configure the PARTID and PMG values within the MPAM information bundle. For the GIC-700, MPAM_NS is fixed to `0b1`, so the GIC-700 is limited to only accessing the Non-secure PARTID space.

While the GIC-700 can generate memory transactions that include an MPAM information bundle, it does not include any MPAM partitionable memory resources. That is, MPAM has no effect on cache allocation or partitioning within the GIC-700.

The Arm Generic Interrupt Controller Architecture Specification describes the GIC MPAM programming requirements.

4. MPAM System Guidance for Infrastructure

This chapter provides general system guidance that can aid designing and implementing an MPAM-enabled system in an infrastructure solution.

4.1 Base System Architecture

Arm processors are built into a large variety of systems. Aspects of this system functionality are crucial to the fundamental function of system software. Variability in PE features, and certain key aspects of the system, impact on the cost of software system development and the associated quality risks.

Base System Architecture (BSA) specifications are part of Arm's strategy of addressing this variability.

To reduce the variability in MPAM systems, BSA specifications include certain requirements for MPAM. At the time of writing, the latest versions of the (BSA) specifications as noted below.

Arm strongly suggests when designing an MPAM-enabled system that the MPAM requirements for the applicable BSA specifications version is followed.

- Arm Base System Architecture 1.0C
- Arm Server Base System Architecture 7.1

4.2 General MPAM System Guidance

This section is inclused guidance on several topics related to system design and integration of an MPAM-enabled system targeting the infrastructure segment.

4.2.1 Number of PARTIDs to support

While there is no definite answer for the amount of PARTIDs a system should support, Arm ecommends that at a minimum, a system will support a unique PARTID per core. This allows a hypervisor to assign a unique PARTID per VM, where a VM may have a core count of one. For example, if we have a 64-core system, the system should support at minimum 64 PARTIDs.

4.2.2 PARTID size

Arm strongly recommends that the PARTID size is the same size in all PEs and MSCs in a system. Software can only use the smallest PARTID size supported by the system. Software convention

typically limits the use to the smallest PARTID in the system. Any larger PARTID spaces are not utilized via software and would become unused real estate within the SoC.

4.2.3 MPAM interrupts

There are two types of interrupts that an MPAM MSC can generate:

- MPAM Error Interrupt
- MPAM Overflow Interrupt

Arm strongly recommends the following integration of MPAM interrupts:

- The MPAM interrupts are implemented as a level-sensitive interrupt.
- The MSC implements separate Secure and Non-secure instances of the MPAM interrupts.
- If the MSC is integrated into a PE, connect the MPAM interrupt signals to the Generic Interrupt Controller (GIC) as either a Private Peripheral Interrupt (PPI) or a Locality-specific Peripheral Interrupt (LPI) for that PE.
- If the MSC is not integrated into a PE, connect the MPAM interrupt signals to the GIC as either a System Peripheral Interrupt (SPI) or LPI.

Only the CMN-700 described in this guide outputs MPAM interrupts. The CMN-700 outputs MPAM Error Interrupts; INTREQMPAMERRNS and INTREQMPAMERRS. As these are not directly integrated into a PE, Arm recommends that these are connected to the GIC as either a SPI or LPI.

4.3 Use of CBusy

CBusy is a three-bit field returned on the response and data channels of CHI-based interconnect. The Completer Busy indication is a mechanism for the Completer of a transaction to indicate its current level of activity. This signaling provides additional information to a Requester on how aggressive it can be in generating speculative activity to improve performance. However, the exact usage of the feedback dependant on the Requester.

An MPAM-enabled memory controller can provide independent CBusy responses for each partition. For example, you canuse a memory controller that supports MPAM bandwidth control to drive CBusy feedback for each partition (PARTID).

Arm does not design or license an MPAM-enabled memory controller. Memory controller MPAM support details will need to be obtained from the relevant memory controller vendor.

4.4 Arm Neoverse Compute Subsystems

Arm Neoverse N2 Compute SubSystem (CSS) is an Arm-configured, verified and performance-validated subsystem of up to 64 N2 cores targeting an advanced 5nm process. Neoverse N2 CSS includes selection and configuration of 3rd Party IP (DDR5 memory and PCIe Gen5/CXL I/O), plus SBSA certification and a platform software model.

The Arm Neoverse N2 Compute SubSystem (CSS) implements the following MPAM-enabled Requesters and MSCs.

- Neoverse N2
- CMN-700
- GIC-700
- MMU-700

Figure 4-1: Arm Neoverse N2 Compute SubSystem (CSS)