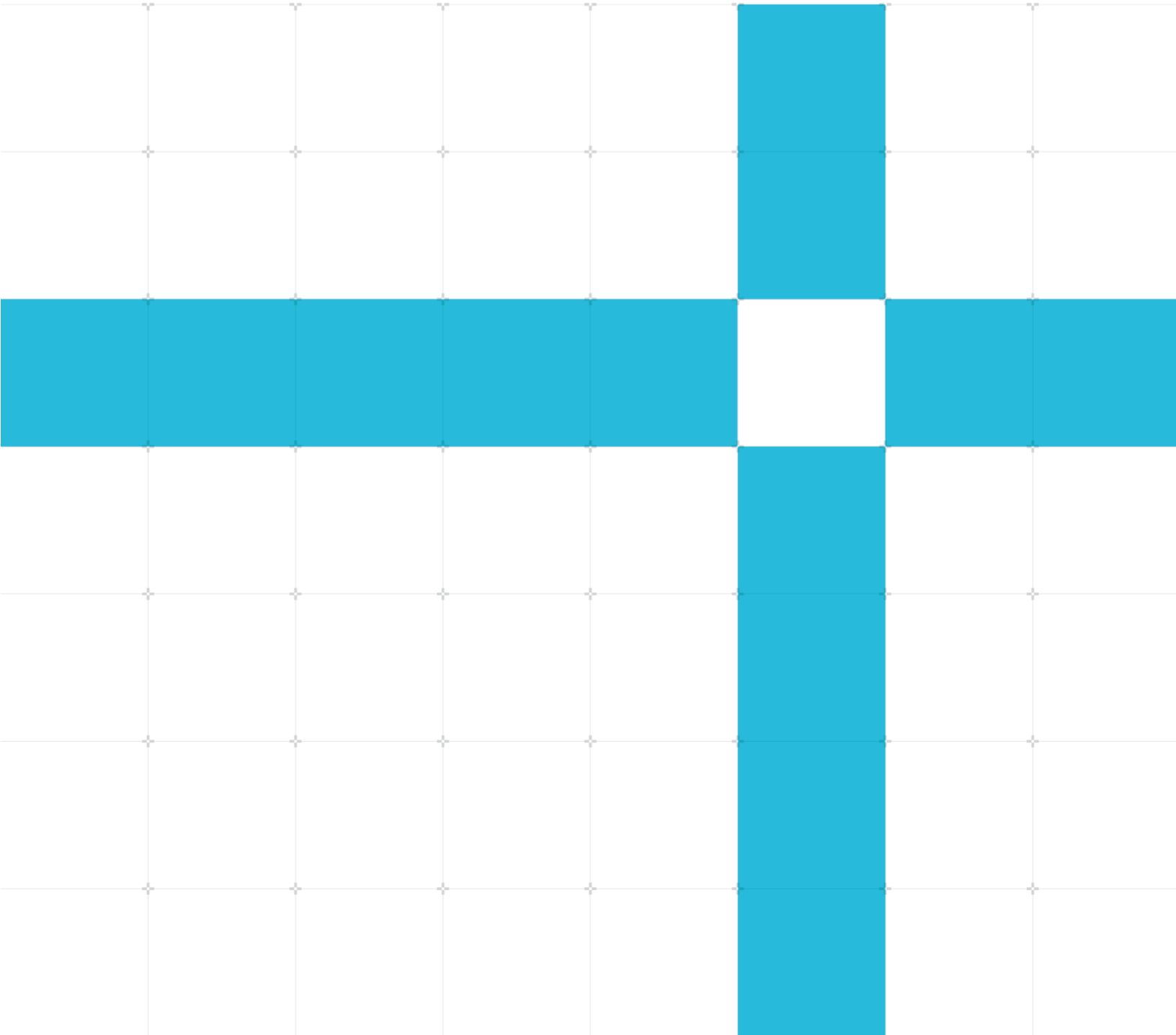
Arm® MPAM Architecture Compliance

Revision: r1p0

Test Scenario

Non-Confidential

Issue 0001-00
ARM040-1254092399-18836



Arm MPAM Test Scenario Document

Release information

Document history

Issue	Date	Confidentiality	Change
01	31 Mar 2025	Non-Confidential	0.5.0 Alpha

Non-Confidential Proprietary Notice

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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 for an Alpha product, that is a product under development.

Progressive terminology commitment

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

This document includes terms that can be offensive. We will replace these terms in a future issue of this document. If you find offensive terms in this document, please email **terms@arm.com**.

Web Address

www.arm.com.

Contents

1 Introduction

1.1 Product revision status

The *rmpn* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

rm Identifies the major revision of the product, for example, r*1*.

pn Identifies the minor revision or modification status of the product, for example, p*2*.

1.2 Intended audience

This document is for engineers who are verifying an implementation of Arm® Memory System Resource Partitioning and Monitoring (MPAM) System Component Architecture.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

1.3.1 Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: **https://developer.arm.com/glossary**.

1.3.2 Typographical Conventions

Convention	Use
italic	Introduces citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` **`bold`**	Denotes language keywords when used outside example code.
`monospace underline`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

1.4 Useful resources

This document contains information that is specific to this product. See the following resources for other relevant information.

- Arm Non-Confidential documents are available on **developer.arm.com/documentation**. Each document link in the tables below provides direct access to the online version of the document.
- Arm Confidential documents are available to licensees only through the product package.

Arm products	Document ID	Confidentiality
MPAM Memory System Component Specification	ARM IHI 0099A.a	Non-Confidential
Arm® Architecture Reference Manual Armv8, for Armv8-A Architecture	ARM DDI 0487L.a	Non-Confidential

1.5 Feedback

Arm welcomes feedback on this product and its documentation.

1.5.1 Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

1.5.2 Feedback on content

If you have comments on content, send an email to **support-systemready-acs@arm.com** and give:

- The title Arm MPAM Test Scenario.
- The number **ARM040-1254092399-18836**
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.
- Arm also welcomes general suggestions for additions and improvements.

2 Arm MPAM MSC Architecture

The MPAM (Memory System Resource Partitioning and Monitoring) architecture is designed for systems running multiple applications or virtual machines on shared memory, particularly in enterprise networking and server environments. It addresses key requirements like controlling the performance impact of non-conforming software, bounding resource usage, and minimizing interference between software components.

MPAM achieves this through two main mechanisms: memory-system resource partitioning and resource usage monitoring. It propagates Partition IDs (PARTID) and Performance Monitoring Groups (PMG) across the memory system, enabling fine-grained control and monitoring of memory resources.

MPAM defines a framework for memory-system components (MSCs) to manage and monitor resource usage, providing standardized, implementation-independent memory-mapped interfaces. These features help align system performance with higher-level software goals and ensure consistent behaviour in complex, multi-tenant environments.

The goal of MPAM ACS is to test the Errors in MSCs, Partitioning & Monitoring related features for compliance against MPAM MSC Specification.

2.1 MPAM ACS

The tests are classified as:

- Register Tests
- Cache Tests
- Error Tests
- Memory Bandwidth Tests

2.2 Register Tests

Test number	Scenario	Algorithm
1	Check MPAM Version EXT Bit Check	• Get the total number of MSC nodes. • Loop through each MSC node by index. • If version is MPAM v1.0, check if EXT field is 0. • If version is MPAM v1.1, check if EXT field is 1. • If EXT field doesn't match the expected value, print error and fail the test. • If version is not 1.0 or 1.1, print error and fail the test. • If all MSCs pass the check, mark the test as passed.
2	Check Expansion of MPAMF_ESR	Get the total number of MSC nodes. • Loop through each MSC node by index. • Read the MPAMF_IDR register value. • If both HAS_ESR and HAS_RIS are set, and EXT is set: • Check if HAS_EXTD_ESR is also set. • If HAS_EXTD_ESR is not set, print error and fail the test. • If all MSCs pass the check, mark the test as passed.

Test number	Scenario	Algorithm
3	Check MPAM MSC Feature Test	Get the total number of MSC nodes. • Loop through each MSC node by index. • Read ID registers: MPAMF_IDR, CCAP_IDR, CSUMON_IDR, MBWUMON_IDR, MSMON_IDR. • If version is MPAM v1.0: • Check that all prohibited fields (e.g., HAS_CMIN, NO_CMAX, HAS_CASSOC, HAS_ENDIS, etc.) are not set. • If any prohibited field is set, print error and increment test failure counter. • If version is MPAM v1.1 or v0.1: • If HAS_IMPL_IDR is set: • Ensure required fields (NO_IMPL_PART, NO_IMPL_MSMON) are also set. • If not, print error and increment failure counter. • Check that EXT field is set. • If not, print error and increment failure counter. • If version is unsupported, print error and fail the test. • After all nodes are checked, if any failure was recorded, mark the test as failed. Otherwise, mark the test as passed.

2.3 Cache Tests

Test number	Scenario	Algorithm
101	Check CPOR Partitioning	• Get the index of the LLC and its cache identifier, Skip the test if LLC info is invalid or not found. • For each MSC node: Find matching cache resource nodes of type PE_CACHE. Check if CPOR and CSU monitor are supported. Track number of CPOR-capable nodes and CSU monitor count. Determine max cache size and min/max PARTID. • Skip the test if CPOR nodes or CSU monitors are not available. • Allocate memory for storing CSU monitor counters for each CPOR scenario. Read and modify MPAM2_EL2 to set PARTID and PMG. • For each MSC node with CPOR support: Configure CPOR Settings and disable CCAP partition settings. Allocate source and destination buffers. Configure and enable CSU monitor. Wait for NRDY timeout to stabilize configuration Capture CSU monitor count before and after memory copy. Store the count difference for analysis. Free allocated buffers. • Restore original MPAM2_EL2 value. • Compare CSU monitor counters across scenarios. If counter increases when it should decrease, mark test as failed. • Set test result to pass if all comparisons are valid, else fail.

Test number	Scenario	Algorithm
102	Check CCAP Partitioning	• Get the index of the LLC and its cache identifier, Skip the test if LLC info is invalid or not found. • For each MSC node: Find matching cache resource nodes of type PE_CACHE. Check if CCAP and CSU monitor are supported. Track number of CCAP-capable nodes and CSU monitor count. • Determine max cache size and min/max PARTID. Skip the test if CCAP nodes or CSU monitors are not available. • Allocate memory for storing CSU monitor counters for each CCAP scenario. Read and modify MPAM2_EL2 to set PARTID and PMG. • For each MSC node with CCAP support: Configure CCAP Settings and disable CPOR partition settings. • Allocate source and destination buffers. • Configure and enable CSU monitor. Wait for NRDY timeout to stabilize configuration. • Capture CSU monitor count before and after memory copy. Store the count difference for analysis. • Free allocated buffers. Restore original MPAM2_EL2 value. • Compare CSU monitor counters across scenarios. If counter increases when it should decrease, mark test as failed. • Set test result to pass if all comparisons are valid, else fail.

Test number	Scenario	Algorithm
103	Check CPOR with CCAP Partitioning	• Get the LLC index and cache identifier; skip the test if either is missing or invalid. • Loop through all MSC nodes and their resources to find PE_CACHE resources that match the LLC identifier and support both CCAP and CPOR; count valid nodes, update cache max size, track CSU monitor count if supported, and update minimum max PARTID value. • Skip the test if there are no CCAP+CPOR nodes or no CSU monitors available. • Allocate a 2D buffer to store CSU monitor counter values for each test scenario and MSC node. • Read and store the current MPAM2_EL2 value, clear the PARTID_D and PMG_D fields, set the desired PARTID and PMG values, and write the updated value back to MPAM2_EL2. • For each enabled test scenario, loop through all MSC nodes, and for each matching PE_CACHE resource, select the resource instance if RIS is supported, configure CCAP and CPOR partitioning according to the scenario (or fallback to 100% if only one is supported), allocate source and destination memory buffers, configure and enable the CSU monitor, wait for NRDY timeout to complete, record the start CSU count, perform memory copy, record the end count, disable the monitor, store the count difference, and free the allocated memory. • Restore the original MPAM2_EL2 register value. • Compare CSU monitor count differences across scenarios for each MSC node; if a later scenario shows more activity than the previous one, log the failure. • Set the final test status as pass if all comparisons succeed or fail if any mismatch is found.

Test number	Scenario	Algorithm
104	Check PMG Storage by CPOR Nodes	• Get the LLC index and cache identifier; skip the test if either is missing or invalid. • Loop through all MSC nodes and their resources to find PE_CACHE resources matching the LLC identifier that support CPOR; for each valid node, configure RIS if supported, update the max cache size, get the max PMG value, count CSU monitors if supported, and track the minimum max PARTID value. • Skip the test if no CPOR-capable nodes or CSU monitors are available. • Loop through MSC nodes again and for each valid CPOR-capable PE_CACHE resource, configure CPOR partitioning to 75% using the selected PARTID. • Set up two PMG groups: PMG1 as default and PMG2 as the highest supported PMG; read and store the current MPAM2_EL2 value, clear PARTID_D and PMG_D fields, set PARTID and PMG1, and write back the modified MPAM2_EL2. • For each matching PE_CACHE resource, allocate source and destination memory buffers sized to 50% of the cache size, log the buffer size, and skip if allocation fails. • Configure CSU monitor with PMG1, enable monitoring, wait for NRDY timeout, perform a memory copy transaction, and record the updated CSU monitor count (storage_value1). • Reconfigure CSU monitor with PMG2, enable monitoring, wait for NRDY timeout again, perform a second memory copy transaction, and record the updated CSU monitor count (storage_value2). • Disable CSU monitor, restore original MPAM2_EL2 value, and free the memory buffers. • Fail the test if storage_value1 is zero (no activity recorded for PMG1) or if storage_value2 is non-zero (unexpected activity for PMG2); otherwise, mark the test as passed.

Test number	Scenario	Algorithm
105	Check PMG Storage by CCAP Nodes	• Get the LLC index and cache identifier; skip the test if either is missing or invalid. • Loop through all MSC nodes and their resources to find PE_CACHE resources matching the LLC identifier that support CCAP; for each valid node, configure RIS if supported, update the max cache size, get the max PMG value, count CSU monitors if supported, and track the minimum max PARTID value. • Skip the test if no CCAP-capable nodes or CSU monitors are available. • Loop through MSC nodes again and for each valid CCAP-capable PE_CACHE resource, configure CCAP partitioning to 75% using the selected PARTID. • Set up two PMG groups: PMG1 as default and PMG2 as the highest supported PMG; read and store the current MPAM2_EL2 value, clear PARTID_D and PMG_D fields, set PARTID and PMG1, and write back the modified MPAM2_EL2. • For each matching PE_CACHE resource, allocate source and destination memory buffers sized to 50% of the cache size, log the buffer size, and skip if allocation fails. • Configure CSU monitor with PMG1, enable monitoring, wait for NRDY timeout, perform a memory copy transaction, and record the updated CSU monitor count (storage_value1). • Reconfigure CSU monitor with PMG2, enable monitoring, wait for NRDY timeout again, perform a second memory copy transaction, and record the updated CSU monitor count (storage_value2). • Disable CSU monitor, restore original MPAM2_EL2 value, and free the memory buffers. • Fail the test if storage_value1 is zero (no activity recorded for PMG1) or if storage_value2 is non-zero (unexpected activity for PMG2); otherwise, mark the test as passed.

Test number	Scenario	Algorithm
106	Check PARTID Storage by CPOR Nodes	• Get the LLC index and cache identifier; skip the test if either is missing or invalid. • Loop through all MSC nodes and their resources to find PE_CACHE resources matching the LLC identifier that support CPOR; for each valid node, configure RIS if supported, update the max cache size, count CSU monitors if supported, and track the number of CPOR-capable nodes. • Skip the test if no CPOR nodes or CSU monitors are found. • Loop through MSC nodes again and for each valid CPOR-capable PE_CACHE resource, configure CPOR partitioning to 75% using the selected test PARTID. • Create two PARTIDs (partid1 and partid2) for generating PE traffic; read and store the current MPAM2_EL2 value, clear PARTID_D and PMG_D fields, set PARTID to partid1 and PMG to default, and write the updated MPAM2_EL2 value. • For each matching PE_CACHE resource, allocate source and destination memory buffers sized to 50% of cache size, log buffer size, and skip if allocation fails. • Configure CSU monitor with partid1, enable monitoring, wait for NRDY timeout, perform a memory transaction, and record updated CSU monitor count as storage_value1. • Reconfigure CSU monitor with partid2, enable monitoring again, wait for NRDY timeout, perform a second memory transaction, and record updated CSU monitor count as storage_value2. • Disable the monitor, restore MPAM2_EL2 to its original value, and free the memory buffers. • Fail the test if storage_value1 is zero (no activity for partid1) or if storage_value2 is non-zero (unexpected activity for partid2); otherwise, mark the test as passed.

Test number	Scenario	Algorithm
107	Check PARTID Storage by CCAP Nodes	• Get the LLC index and cache identifier; skip the test if either is missing or invalid. • Loop through all MSC nodes and their resources to find PE_CACHE resources matching the LLC identifier that support CCAP; for each valid node, configure RIS if supported, update the max cache size, count CSU monitors if supported, and track the number of CCAP-capable nodes. • Skip the test if no CCAP nodes or CSU monitors are found. • Loop through MSC nodes again and for each valid CCAP-capable PE_CACHE resource, configure CCAP partitioning to 75% using the selected test PARTID. • Create two PARTIDs (partid1 and partid2) for generating PE traffic; read and store the current MPAM2_EL2 value, clear PARTID_D and PMG_D fields, set PARTID to partid1 and PMG to default, and write the updated MPAM2_EL2 value. • For each matching PE_CACHE resource, allocate source and destination memory buffers sized to 50% of cache size, log buffer size, and skip if allocation fails. • Configure CSU monitor with partid1, enable monitoring, wait for NRDY timeout, perform a memory transaction, and record updated CSU monitor count as storage_value1. • Reconfigure CSU monitor with partid2, enable monitoring again, wait for NRDY timeout, perform a second memory transaction, and record updated CSU monitor count as storage_value2. • Disable the monitor, restore MPAM2_EL2 to its original value, and free the memory buffers. • Fail the test if storage_value1 is zero (no activity for partid1) or if storage_value2 is non-zero (unexpected activity for partid2); otherwise, mark the test as passed.

2.4 Error Tests

Test number	Scenario	Algorithm
201	Check MSC PARTID selection range error	• Get PE index and total MSC node count. • Loop through each MSC; skip if MPAMF_ESR is not supported. • Save current MPAMF_ECR value for later restore. Reset error code; fail test if reset fails. • Program MPAMCFG_PART_SEL with out-of-range PARTID. Wait briefly for error to propagate. • Read MPAMF_ESR and check for expected error code. If error code mismatch, log and mark test as failed. • Restore original MPAMF_ECR value. • After loop, skip test if no ESR-capable nodes were found; fail if any mismatch; else pass.
202	Check request PARTID out-of-range error	• Get PE index, total MSC count, and save MPAM2_EL2. • Loop through MSCs; skip if MPAMF_ESR not supported. Save MPAMF_ECR and reset error code; fail if reset fails. • Program MPAM2_EL2 with PARTID that is beyond the range of MSC's maximum PARTID; if skipped, restore MPAM2_EL2 and continue. Mark test as executed. • Wait briefly, read MPAMF_ESR, and check for expected error code; fail if mismatched. • Restore MPAM2_EL2 and MPAMF_ECR. Mark test as skipped, failed, or passed based on results.
203	Check MSMON config ID out-of-range error	• Get PE index and total number of MSC nodes. • Loop through MSCs; skip if MPAMF_ESR is not supported. Save MPAMF_ECR and reset any existing error code; fail if reset fails. • Check if the MSC supports monitors and get the total count of CSU and MBWU monitors. • Skip if no monitors are present in the MSC. Mark that at least one MSC ran the test. • Generate a monitor config ID out-of-range error using the monitor count. Wait briefly, read MPAMF_ESR, and verify the error code; fail if it doesn't match expected. • Restore original MPAMF_ECR value. Mark test as skipped if no eligible MSCs, failed if any mismatch, otherwise passed.

Test number	Scenario	Algorithm
204	Check request PMG out-of-range error	• Get PE index and total MSC count; save current MPAM2_EL2. • Loop through MSCs; skip if MPAMF_ESR not supported. Save MPAMF_ECR and reset error code; fail if reset fails. • Program MPAM2_EL2 with PMG that is beyond the range of MSC's maximum PMG; if skipped, restore MPAM2_EL2 and continue. • Mark test as executed; wait, read MPAMF_ESR, and compare with expected error code. • Fail if error code mismatches; restore MPAM2_EL2 and MPAMF_ECR. Set final status as skipped, failed, or passed based on test outcome.
205	Check MSC MONITOR selection range error	• Get PE index and total MSC count. Loop through MSCs; skip if ESR not supported or no monitors are implemented. • Count CSU and MBWU monitors; skip MSC if none found. • Save MPAMF_ECR and reset error code; fail if reset fails. Mark test as executed, generate Monitor selection range error using total monitor count. • Wait, read MPAMF_ESR, and check for expected error code; fail if mismatched. • Restore MPAMF_ECR and update final test status as skipped, failed, or passed.
206	Check intPARTID out-of-range error	• Get PE index and total MSC node count. Loop through MSCs; skip if MPAMF_ESR or PARTID narrowing not supported. • Save MPAMF_ECR and reset error code; fail if reset fails. Mark test as executed, get max PARTID and max internal PARTID. • Write a valid reqPARTID and map it to an out-of-range intPARTID (max intPARTID + 1). • Wait, read MPAMF_ESR, and verify expected error code; fail if mismatched. • Restore MPAMF_ECR and set final test result as skipped, failed, or passed.

Test number	Scenario	Algorithm
207	Check Unexpected internal error	• Get PE index and total number of MSCs. • Loop through MSCs; skip if MPAMF_ESR or PARTID narrowing is not supported. Save MPAMF_ECR and reset error code; fail if reset fails. • Mark test as executed, get max PARTID and max internal PARTID. • Configure MPAMCFG_PART_SEL to select internal mapping using a valid PARTID. • Configure MPAMCFG_INTPARTID with the max internal PARTID to trigger an unexpected internal error. • Wait, read MPAMF_ESR, and check for expected error code; fail if mismatched. Restore MPAMF_ECR and set final status as skipped, failed, or passed.
208	Check undefined RIS in MPAMCFG_PART_SEL error	• Get PE index and total number of MSCs. • Loop through MSCs; skip if MPAMF_ESR or RIS is not supported. Save MPAMF_ECR and reset error code; fail if reset fails. • Get max PARTID and max RIS index, then write an out-of-range RIS to MPAMCFG_PART_SEL. • Trigger error by accessing the same register; wait, then read MPAMF_ESR. • If expected RIS error code is not seen, check if MPAMCFG_PART_SEL behaves as RAZ/WI; fail if not. • Restore MPAMF_ECR, and mark the test as skipped, failed, or passed based on result.
209	Check RIS no control error	• Get PE index and total MSC node count. Loop through MSCs; skip if MPAMF_ESR or RIS is not supported. • For each resource in the MSC, save MPAMF_ECR and reset error code; fail if reset fails. Select the current RIS index using MPAMCFG_PART_SEL.RIS. • Check if CCAP, CPOR, or MBW partitioning is unimplemented; if so, write to the corresponding partition register to trigger an error. • Read MPAMF_ESR and verify if the RIS_NO_CTRL error code is reported. • If error code mismatches, check if the register behaves as RAZ/WI; fail if not. • Restore MPAMF_ECR and update final test status as skipped, failed, or passed.

Test number	Scenario	Algorithm
210	Check undefined RIS in MSMON_CFG_MON_SEL error	• Get PE index and total MSC count. Loop through MSCs; skip if MPAMF_ESR, RIS, or MSMON is not supported. • Save MPAMF_ECR and reset error code; fail if reset fails. Get max RIS index and write out-of-range RIS to MSMON_CFG_MON_SEL. • Access the same register to trigger the error; wait and read MPAMF_ESR. • If expected error code is not reported, verify if register behaves as RAZ/WI; fail if not. • Restore MPAMF_ECR and mark test as skipped, failed, or passed.
211	Check RIS no monitor error	• Get PE index and total number of MSCs. Loop through MSCs; skip if MPAMF_ESR or RIS is not supported. • For each resource in the MSC, save MPAMF_ECR and reset error code; fail if reset fails. Configure MSMON_CFG_MON_SEL.RIS to select current resource. • If CSU or MBWU monitoring is not implemented, write to the corresponding unimplemented MSMON_CFG register to trigger an error. • Read MPAMF_ESR and check for RIS_NO_MON error; if missing, check if register behaves as RAZ/WI and fail if not. • Restore MPAMF_ECR; mark the test as skipped, failed, or passed based on results.
212	Check MSC Level-Sensitive Error Interrupt Behaviour	• Get PE index and total MSC count. Loop through MSCs; skip if MPAMF_ESR is not supported. • Get error interrupt number and type; skip if interrupt is not level-triggered or not implemented. • Save MPAMF_ECR and reset error code; fail if reset fails. Register the interrupt handler and route the interrupt to the PE. • Trigger an MSC error interrupt by writing nonzero value to MPAMF_ESR. Poll until interrupt is received or timeout occurs. • Restore MPAMF_ECR; fail if interrupt not received in time. • Mark test as skipped if no MSC supports error interrupt; otherwise, pass or fail based on result.

Test number	Scenario	Algorithm
213	Check MSC Edge-Trigger Error Interrupt Behaviour	• Get PE index and total MSC count. Loop through MSCs; skip if MPAMF_ESR not supported. • Get error interrupt number and flags; skip if not edge-triggered or not implemented. • Save MPAMF_ECR and reset error code; fail if reset fails. Register interrupt handler and route the interrupt to the PE. • Try to trigger an MSC error interrupt via MPAMF_ESR. Poll for interrupt completion; if handler catches interrupt in time, fail the test. • Restore MPAMF_ECR and mark test as passed, failed, or skipped based on result.
214	Check MBWU monitor overflow interrupt functionality	• Get PE index and total MSC count, save and update MPAM2_EL2 with default PARTID and PMG. • Loop through MSCs and their resources; check for RIS, MBWU monitor, and overflow interrupt support. • Skip if monitor or interrupt not supported; otherwise, register interrupt handler and route it to PE. Trigger MBWU overflow by enabling monitor, writing the maximum supported counter value to MBWU monitor and performing large memory copy using 10MB buffers. • Wait for NRDY timeout, perform memory copy, and poll for interrupt completion. • If interrupt not received in time, fail the test; otherwise, reset status and free buffers. Restore MPAM2_EL2, disable and reset monitor after test. • Mark test as skipped if no MSC supports overflow interrupt; else pass or fail based on result.

2.5 Memory Bandwidth Tests

Test number	Scenario	Algorithm
301	Check MBWPBM Partitioning	• Get PE index and total MSC count; identify memory resources supporting MBWPBM and track minmax PARTID. • Skip test if no MBWPBM-supported memory nodes exist. Disable CPOR and CCAP partitioning across all MSCs for minmax PARTID. • Update MPAM2_EL2 with minmax PARTID and default PMG and prepare a counter array to log MBWU activity. • For each test scenario (e.g., 20% and 5% partition): Loop through memory MSC nodes, if MBWPBM is supported: • Configure MBWPBM for the scenario and disable MBWMIN/MBWMAX if present. Allocate source and destination buffers based on available address range and required size. • Skip if SRAT info is missing or memory cannot be allocated. If MBWU monitor is not found, skip the test. • Configure and enable the MBWU monitor, then perform memory copy. Record start and end MBWU monitor counts to measure bandwidth usage. • Reset and disable MBWU monitor and free allocated memory. Restore original MPAM2_EL2 settings. • Compare recorded monitor counts across scenarios; if higher bandwidth usage is recorded in more restricted settings, mark test as failed. Otherwise, mark test as passed.

Test number	Scenario	Algorithm
302	Check MBWMIN Partitioning	• Get PE index and total MSC count; identify memory nodes supporting MBWMIN and find the minimum valid PARTID. Skip the test if no MBWMIN-capable memory nodes exist. • Set up MPAM2_EL2 with default PMG and minmax PARTID, and configure all cache nodes to disable CPOR, CCAP, MBWPBM, and MBWMAX. Install exception handlers for sync and async faults. • For each MBWMIN-capable memory node: Configure RIS, disable MBWPBM and MBWMAX. • Allocate shared buffers and determine buffer size. Launch contention by spawning memory-copy threads on secondary PEs. • Scenario 1: Configure MBWMIN with BW1 percentage and record MBWU monitor count. Wait for all secondary PEs to complete; if timeout, fail the test. • Scenario 2: Reconfigure MBWMIN with BW2 percentage and repeat measurement. Compare bandwidth usage across scenarios; fail if BW usage increases when a stricter limit is set. • Restore MPAM2_EL2 and clean up memory allocations. Mark test as passed if all conditions are met.
303	Check MBWMAX Partitioning	• Get PE index and total MSC count; identify memory resources supporting MBWMAX and track minmax PARTID. • Skip test if no MBWMAX-supported memory nodes exist. Disable CPOR and CCAP partitioning across all MSCs for minmax PARTID. • Update MPAM2_EL2 with minmax PARTID and default PMG and prepare a counter array to log MBWU activity. • For each test scenario (e.g., 20% and 5% partition): Loop through memory MSC nodes; if MBWMAX is supported: Configure MBWMAX for the scenario and disable MBWMIN/MBWPBM if present. • Allocate source and destination buffers based on available memory and required size. Skip if SRAT info is invalid or buffer allocation fails. • If MBWU monitor is not found, skip the test. Configure and enable the MBWU monitor, then perform memory copy. • Record start and end MBWU monitor counts to measure bandwidth usage. Reset and disable MBWU monitor and free allocated memory. Restore original MPAM2_EL2 settings. • Compare recorded monitor counts across scenarios; if usage is higher at a more restrictive setting, mark test as failed. Otherwise, mark test as passed.

Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue 01

Change	Location
First release	-