Arm® MPAM Architecture Compliance Test Scenario

Version 1.0

Arm® MPAM Architecture Compliance

Test Scenario

Release Information

Document History

Version	Date	Confidentiality	Change
1.0	13 July 2018	Non-Confidential	First version of the document.

Confidentiality Status

Product Status

The information in this document is beta, that is for a product under development.

Web Address

http://www.arm.com

Contents

1 About this document

This document describes the test scenarios for MPAM architecture compliance.

1.1. References

Reference	Document	Author	Title
1	ARM-ECM-0485919	Arm	MPAM Extension Architecture v1.0

1.2. Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
ACS	Arm Architecture Compliance Suite – A group of architecture tests for the verification of an architecture feature set.
CCAP	Cache Capacity Partitioning
CPOR	Cache Portion Partitioning
CSUMON	Cache Storage Usage Monitor
Level	Cache Level (L1-L7)
MBW	Memory Bandwidth
MBWMAX	Memory Bandwidth Maximum Limit Partitioning
MBWMIN	Memory Bandwidth Minimum Limit Partitioning
MBWPOR	Memory Bandwidth Portion Partitioning
MBWUMON	Memory Bandwidth Usage Monitor
MPAM	Memory Partitioning And Monitoring
MSC	Memory System Component
PARTID	Partitioning Identifier
PE	Processing Element
PMG	Performance Monitoring Group

1.3. Scope

This document describes the verification scenarios and the strategy that is followed for creating ACS tests for configuration APIs, as described in the MPAM architecture.

2 Introduction

MPAM specification provides optional additions to the Arm architecture to support memory system partitioning and monitoring facilities that measure the resource usage by software. This document defines standard APIs that are vendor-neutral, interoperable, and software portable.

The goal of the compliance suite is to the test the partitioning and monitoring features of cache and memory nodes for compliance against the MPAM specification.

3 Cross reference to architecture and tests

The following table shows the cross references from the architecture specification to the corresponding scenarios in the MPAM ACS.

Test number	Specification section	Test scenario
1	3.0	MPAM aware system
2	12.2.1	Cache portion partitioning
3	12.2.2	Cache maximum capacity partitioning
4	12.2.2.3	Cache capacity partitioning with cache portion partitioning
5	7.0	PARTID storage by cache portion partitioning nodes
6	7.0	PARTID storage by cache capacity partitioning nodes
7	7.0	PMG storage by cache portion partitioning monitors
8	7.0	PMG storage by cache capacity partitioning monitors
9	12.3.1	New monitor configuration does not affect ongoing monitor
10	11.7.1.1	MSC level-sensitive error interrupt behavior
11	11.7.1.2	MSC edge-trigger error interrupt behavior
12	15.1.2	MSC PARTID selection range error
13	15.1.6	MSC monitor selection range error
14	15.1.3	Request PARTID out of range error
15	15.1.5	Request PMG out of range error
16	15.1.4	MSMON configuration ID out-of-range error
17	12.2.3	Memory bandwidth portion partitioning
18	12.2.4.1.1	Minimum bandwidth limit partitioning
19	12.2.4.1.2	Maximum bandwidth limit partitioning
20	14.5.5	Memory system monitor overflow interrupt functionality

The following table shows the cross references to the architecture specification to the corresponding scenarios that are not implemented in this version of MPAM ACS.

Specification section	Test scenario
7.0	Cache eviction must attach MPAM fields to the request to store the evicted data. The source for MPAM information on an eviction may depend on whether eviction is to memory or to another cache.
7.7.1	MPAM information on a request to the cache from an upstream master is used to provide MPAM information for downstream requests to fulfil the incoming requests such as a read from downstream on a cache miss that fetches data to the cache.
7.7.1	MPAM information on a request to the cache from an upstream master is used to optionally provide MPAM information for downstream requests generated by evict or clean operations when this cache is the last level of cache upstream of main memory.
7.7.1	MPAM information on a request to the cache from an upstream master is used for triggering and filtering performance monitors if implemented for events triggered by requests from upstream masters.
7.7.1	MPAM information is used to provide the MPAM information for downstream requests generated by evict or clean operations when this is not the last level of cache.
7.7.1	MPAM information is used for triggering and filtering performance counters by MPAM partition and performance monitoring group if implemented for events triggered by internal and downstream requests.
7.7.2	A cache may optionally produce the MPAM information of the request that caused the eviction in its request to a memory channel controller or produce the stored MPAM information associated with the evicted line.

4 Verification scenarios

All the scenarios that are described in this section can be tested from the UEFI shell application.

4.1. MPAM aware system

Test # 1

Steps:

1. Read MPAM ACPI table for the availability of MSC partitioning nodes.

4.2. Cache portion partitioning

Test # 2

Steps:

1. Read the maximum partition ID supported by each MPAM cache node from all cache levels.
2. Find the minimum among those partition IDs.
3. Update the current PE with the partition ID and PMG values.
4. Create a portion partition associated with above determined partition ID and for each cache node equivalent to 3/4th of its cache size.
5. Disable capacity partitioning for all levels of cache nodes.
6. Enable portion partitioning for all levels of cache nodes.
7. Perform a memcopy of size equal to 3/4th of the maximum cache size and measure the copy latency.
8. Free the memcopy buffers allocated in step 7 and obtain new buffers from the heap manager.
9. Change the portion partition size to 1/4th of cache size and repeat step 7.
10. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.3. Cache maximum capacity partitioning

Test # 3

Steps:

1. Read the maximum partition ID supported by each MPAM cache node from all cache levels.
2. Find the minimum among those partition IDs.
3. Update the current PE with the partition ID and PMG values.
4. Create a capacity partition for each cache node equivalent to 3/4th of its cache size. Associate the above configured partition ID with these partitions.
5. Disable portion partitioning for all levels of cache nodes.
6. Enable capacity partitioning for all levels of cache nodes.
7. Perform a memcopy of size equal to 3/4th of the maximum cache size and measure the copy latency.
8. Free the memcopy buffers used in step 7 and obtain new buffers from the heap manager.
9. Change the capacity partition size to 1/4th of cache size and repeat step 7.
10. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.4. Cache capacity partitioning with cache portion partitioning

Test # 4

Steps:

1. Read the maximum partition ID supported by each MPAM cache node from all cache levels.
2. Find the minimum among those partition IDs.
3. Update the current PE with the partition ID and PMG values.
4. Create a capacity partition for each cache node equivalent to 3/4th of its cache size and associate the above partition ID with each of these capacity partitions.
5. Within the above capacity partition of each cache node, create a portion partition of equivalent size as capacity partition.
6. Enable capacity partitioning for all levels of cache nodes.
7. Enable portion partitioning for all levels of cache nodes.
8. Perform a memcopy of size equal to 3/4th of the max cache size and measure the copy latency.
9. Free the memcopy buffers used in step 8 and obtain new buffers from the heap manager.
10. Keep the capacity partition size same, change the portion partition size to 1/4th of cache size and repeat step 8.
11. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.5. PARTID storage by cache portion partitioning nodes

Test # 5

Steps:

1. Read the maximum partition ID supported by each MPAM cache node from all cache levels.
2. Find the minimum among those partition IDs.
3. Create a portion partition 1 for each cache node equivalent to 3/4th of its cache size. Associate the above part ID with this partition.
4. Create portion partition 2 for each cache node equivalent to 3/4th of its cache size and associate the above 'PARTID – 1' with this partition.
5. Disable capacity partitioning for all levels of cache nodes.
6. Enable portion partitioning for all levels of cache nodes.
7. Update PE with the partition ID and PMG values.
8. Perform a memcopy of size equal to 3/4th of the max cache size and measure the copy latency.
9. Write (minmax PARTID - 1, default PMG) to the PE register MPAM2_EL2 to generate partitioning 2 control information to the downstream MSC components.
10. Do not free the memcopy buffers used in step 8 and use the same buffers to perform step 11.
11. Perform another memcopy of size same as in step 7 and measure the copy latency.
12. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.6. PARTID storage by cache capacity partitioning nodes

Test # 6

Steps:

1. Read the maximum partition ID supported by each MPAM cache node from all cache levels and find the minimum among those partition IDs.
2. Create a capacity partition 1 for each cache node equivalent to 3/4th of its cache size. Associate minmax PARTID with these portion partitions.
3. Create capacity partition 2 for each cache node equivalent to 3/4th of its cache size and associate (minmax PARTID -1) with these portion partitions.
4. Disable portion partitioning for all levels of cache nodes for partition 1 designated with minmax PARTID and for partition 2 designated with (minmax PARTID – 1).
5. Enable capacity partitioning for all levels of cache nodes for partition 1 designated with minmax PARTID and for partition 2 designated with (minmax PARTID -1).
6. Write (minmax PARTID, default PMG) to the PE register MPAM2_EL2 to generate partition 1 control information to the downstream MSC components.
7. Perform a memcopy of size equal to 3/4th of the maximum cache size and measure the copy latency.
8. Write (minmax PARTID - 1, default PMG) to the PE register MPAM2_EL2 to generate partition 2 control information to the downstream MSC components.
9. Do not free the memcopy buffers used in step 7 and use the same buffers to perform step 10.
10. Perform another memcopy of size same as in step 7 and measure the copy latency.
11. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.7. PMG storage by cache portion partitioning monitors

Test # 7

Steps:

1. Read the maximum PMG supported by each MPAM cache node from all cache levels and find the minimum among those PMGs.
2. Create a portion partition 1 for each cache node equivalent to 3/4th of its cache size. Associate minmax PMG with these partitions.
3. Create portion partition 2 for each cache node equivalent to 3/4th of its cache size and associate (minmax PMG -1) with these partitions.
4. Disable capacity partitioning for all levels of cache nodes for partition 1 designated with minmax PMG and for partition 2 designated with (minmax PMG – 1).
5. Enable portion partitioning for all levels of cache nodes for partition 1 designated with minmax PMG and for partition 2 designated with (minmax PMG -1).
6. Write (default PARTID, minmax PMG) to the PE register MPAM2_EL2 to generate partition 1 control information to the downstream MSC components.
7. Perform a memcopy of size equal to 3/4th of the maximum cache size and measure the copy latency.
8. Write (default PARTID, minmax PMG - 1) to the PE register MPAM2_EL2 to generate partition 2 control information to the downstream MSC components.
9. Do not free the memcopy buffers used in step 7 and use the same buffers to perform step 10.
10. Perform another memcopy of size same as in step 7 and measure the copy latency.
11. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.8. PMG storage by cache capacity partitioning monitors

Test # 8

Steps:

1. Read the maximum PMG supported by each MPAM cache node from all cache levels and find the minimum among those PMGs.
2. Create a capacity partition 1 for each cache node equivalent to 3/4th of its cache size. Associate minmax PMG with these partitions.
3. Create capacity partition 2 for each cache node equivalent to 3/4th of its cache size and associate (minmax PMG -1) with these partitions.
4. Disable portion partitioning for all levels of cache nodes for partition 1 designated with minmax PMG and for partition 2 designated with (minmax PMG – 1).
5. Enable capacity partitioning for all levels of cache nodes for partition 1 designated with minmax PMG and for partition 2 designated with (minmax PMG -1).
6. Write (default PARTID, minmax PMG) to the PE register MPAM2_EL2 to generate partition 1 control information to the downstream MSC components.
7. Perform a memcopy of size equal to 3/4th of the maximum cache size and measure the copy latency.
8. Write (default PARTID, minmax PMG - 1) to the PE register MPAM2_EL2 to generate partition 2 control information to the downstream MSC components.
9. Do not free the memcopy buffers used in step 7 and use the same buffers to perform step 10.
10. Perform another memcopy of size same as in step 7 and measure the copy latency.
11. Test passes if memory copy latency in scenario 2 is more than the latency in scenario 1.

4.9. New cache monitor configuration does not affect ongoing cache monitor

Test # 9

Steps:

1. Read the maximum PMG supported by each MPAM cache node from all cache levels and find the minimum among those PMG values.
2. Create monitor 1 for any cache node that supports monitoring. Associate default PARTID and minmax PMG with this monitor.
3. Disable portion and capacity partitioning for all other cache nodes for default PARTID.
4. Enable portion and capacity partitioning for this cache node for default PARTID.
5. Write (default PARTID, minmax PMG) to the PE register MPAM2_EL2 to generate partitioning control information to the downstream MSC components.
6. Perform a memcopy of size equal to 3/4th of the maximum cache size.
7. Create monitor 2 for the same cache node. Associate default PARTID and (minmax PMG -1) with this second monitor.
8. Perform memcopy of size equal to 3/4th of the max cache size.
9. Repeat the above steps for all cache nodes and for all monitors of each cache node.
10. Test passes if CSU monitor value in scenario 2 is comparable to scenario 1 for all cache nodes and all monitors.

4.10. MSC level-sensitive error interrupt behavior

Test # 10

Steps:

1. Read ACPI tables and obtain the error interrupt number and type of each MSC.
2. Register an interrupt handler.
3. Set the interrupt enable bit in MPAM_ECR and generate an interrupt by writing nonzero to MPAMF_ESR.
4. Test passes if software can make a level-sensitive interrupt active by writing nonzero to MPAMF_ESR.
5. Repeat the above steps to verify the level-sensitive interrupt behavior for all MSC.

Note: If the MSCs contain no error detection or cannot guarantee any errors due to software writes, the partner can submit this explanation for the test failure and claim a waiver for the test.

4.11. MSC edge-triggered error interrupt behavior

Test # 11

Steps:

1. Read ACPI tables and obtain the error interrupt number and type of each MSC.
2. Register an interrupt handler for the above error interrupt.
3. Set the interrupt enable bit in MPAM_ECR and try to generate an interrupt by writing nonzero to MPAMF_ESR.
4. Test passes if edge-triggered interrupt is not generated when software writes to MPAMF_ESR.
5. Repeat the above steps to verify the edge-triggered interrupt behavior for all MSC.

4.12. MSC PARTID selection range error

Test # 12

Steps:

1. Read ACPI tables and obtain the error interrupt number of each MSC.
2. Register an interrupt handler for the above error interrupt.
3. Set the interrupt enable bit in MPAM_ECR and try to generate an MPAM hardware error by writing an out-of-range PARTID (max PARTID + 1) to MPAMCFG_PART_SEL.PARTID_SEL field.
4. Test passes if the PARTID out-of-range error interrupt is generated when software writes to MPAMCFG_PART_SEL.PARTID_SEL field.
5. Repeat the above steps to verify the PARTID selection range error behavior for all MSC.

Note: If the MSCs are not able to detect this error due to being unable to truncate the PARTID to thee implemented width, the partner can submit this explanation for test failure and claim a waiver for the test.

4.13. MSC monitor selection range error

Test # 12

Steps:

1. Read ACPI tables and obtain the error interrupt number of each MSC monitor.
2. Register an interrupt handler for the above error interrupt.
3. Set the interrupt enable bit in MPAM_ECR and try to generate a monitor hardware error by writing an out-of-range monitor (monitor count + 1) value to MSMON_CFG_MON_SEL.MON_SEL field.
4. Test passes if the monitor out-of-range error interrupt is generated when software writes to MSMON_CFG_MON_SEL.MON_SEL field.

5. Repeat the above steps to verify the monitor selection range error behavior for all MSC monitors.

Note: If the MSCs are not able to detect this error due to being able to truncate the out-of-range PMG to an in-range PMG, the partner can submit this explanation for test failure and claim a waiver for the test.

4.14. Request PARTID out-of-range error

Test # 12

Steps:

1. Read ACPI tables and obtain the error interrupt number of each MSC.
2. Register an interrupt handler for the above error interrupt.
3. Extract maximum PARTID supported by this MSC MPAMF_IDR and call this MSC max PARTID.
4. Extract maximum PARTID supported by MPAMIDR_EL1 and call this PE max PARTID.
5. Skip this test if MSC max PARTID exceeds PE max PARTID.
6. Set the interrupt enable bit in MPAM_ECR and try to generate an MPAM hardware error by writing MSC out-of-range PARTID (MSC max PARTID + 1) to MPAM2_EL2.
7. Test passes if the request PARTID out-of-range error interrupt is generated when software writes to MPAM2_EL2.
8. Repeat the above steps to verify the request PARTID out-of-range error behavior for all MSC.

Note: If the CPU implementation truncates an out-of-range PARTID to an in-range PARTID, the partner can submit this explanation for the test failure and claim a waiver for the test.

4.15. Request PMG out-of-range error

Test # 13

Steps:

1. Read ACPI tables and obtain the error interrupt number of each MSC.
2. Register an interrupt handler for the above error interrupt.
3. Extract max PMG supported by this MSC MPAMF_IDR and call this MSC max PMG.
4. Extract max PMG supported by MPAMIDR_EL1 and call this PE max PMG.
5. Skip this test if MSC max PMG exceeds PE max PMG.
6. Set the interrupt enable bit in MPAM_ECR and try to generate an MPAM hardware error by writing MSC out-of-range PMG (MSC max PMG + 1) to MPAM2_EL2.
7. Test passes if the request PMG out-of-range error interrupt is generated when software writes to MPAM2_EL2.
8. Repeat the above steps to verify the request PMG out-of-range behavior for all MSC.

Note: If the CPU implementation truncates an out-of-range PMG to an in-range PMG, the partner can submit this explanation for the test failure and claim a waiver for the test.

4.16. MSMON configuration ID out-of-range error

Test # 13

Steps:

1. Read ACPI tables and obtain the error interrupt number of each Memory node.
2. Register an interrupt handler for the above error interrupt.
3. Extract max PARTID supported by this memory node from its MPAMF_IDR and call this MSC max PARTID.

4. Set the interrupt enable bit in MPAM_ECR and try to generate an MPAM hardware error by writing out-of-range PARTID (MSC max PARTID + 1) to the memory node monitor.
5. Test passes if the MSMON configuration ID out-of-range error interrupt is generated when software writes out-of-range PARTID to the monitor configuration register.
6. Repeat the above steps to verify MSMON config ID out-of-range behavior for all memory monitors.

4.17. Memory bandwidth portion partitioning

Test # 14

Steps:

1. Read max PARTID supported by each MSC and derive the minmax PARTID.
2. Write (minmax PARTID, default PMG) to the PE register MPAM2_EL2 to generate partitioning control information to the downstream MSC components.
3. Disable portion and capacity partitioning for all levels of cache nodes for a partition designated with minmax PARTID.
4. Create a memory portion partition for each memory node equivalent to 3/4th of its size. Associate minmax PARTID with these partitions.
5. Disable min and max bandwidth partitioning for all memory nodes for a partition designated with minmax PARTID.
6. Perform a memcopy of size 1MB and measure the copy latency.
7. Change the memory portion partition size to 1/4th of its size and repeat step 6.
8. Test passes if memcopy latency in scenario 2 is more than scenario 1.
9. Repeat the above procedure for all the MPAM memory nodes.

4.18. Minimum bandwidth limit partitioning

Test # 15

Steps:

1. Read max PARTID supported by each MSC and derive the minmax PARTID.
2. Write (minmax PARTID, default PMG) to the PE register MPAM2_EL2 to generate partitioning control information to the downstream MSC components.
3. Disable portion and capacity partitioning for all levels of cache nodes for a partition designated with minmax PARTID.
4. Disable portion and max bandwidth limit partitioning for this memory node for a partition designated with minmax PARTID.
5. Create a minimum bandwidth limit partitioning for this memory node equivalent to 1/4th of its size. Associate minmax PARTID with this partition.
6. Create bandwidth contention with secondary PE generated traffic and perform a memcopy of size 256MB on the primary PE and measure the copy latency.
7. Change the minimum bandwidth limit to 3/4th of its size and repeat step 6.
8. Test passes if memory copy latency in scenario 2 is less than scenario 1.

4.19. Maximum bandwidth limit partitioning

Test # 16

Steps:

1. Read max PARTID supported by each MSC and derive the minmax PARTID.

2. Write (minmax PARTID, default PMG) to the PE register MPAM2_EL2 to generate partition control information to the downstream MSC components.
3. Disable portion and capacity partitioning for all levels of cache nodes for a partition designated with minmax PARTID.
4. Disable portion and min bandwidth limit partitioning for this memory node for a partition designated with minmax PARTID.
5. Create a maximum bandwidth limit partitioning for this memory node equivalent to 1/4th of its size. Associate minmax PARTID with this partition.
6. Create bandwidth contention with secondary PE generated traffic and perform a memcopy of size 256MB on the primary PE and measure the copy latency.
7. Change the maximum bandwidth limit to 3/4th of its size and repeat step 6.
8. Test passes if memory copy latency in scenario 2 is less than the latency in scenario 1.

4.20. MSMON overflow interrupt functionality

Test # 16

Steps:

1. Read ACPI tables and obtain the overflow interrupt number of each memory node.
2. Register an interrupt handler for the above overflow interrupt.
3. Set the interrupt enable bit in MPAM_ECR and enable the overflow interrupt by setting OFLOW_INTR bit in MSMON_CFG_MBWU_CTL register.
4. Write the maximum memory bandwidth usage value to MSMON_MBWU register and perform an additional memcopy of size 2MB to cause the overflow interrupt by the MPAM hardware.
5. Test passes if the memcopy generates an overflow interrupt.
6. Repeat the above steps to verify the overflow interrupt functionality for all memory nodes.