CoreLink MMU-600AE System Memory Management Unit

Software Developer Errata Notice

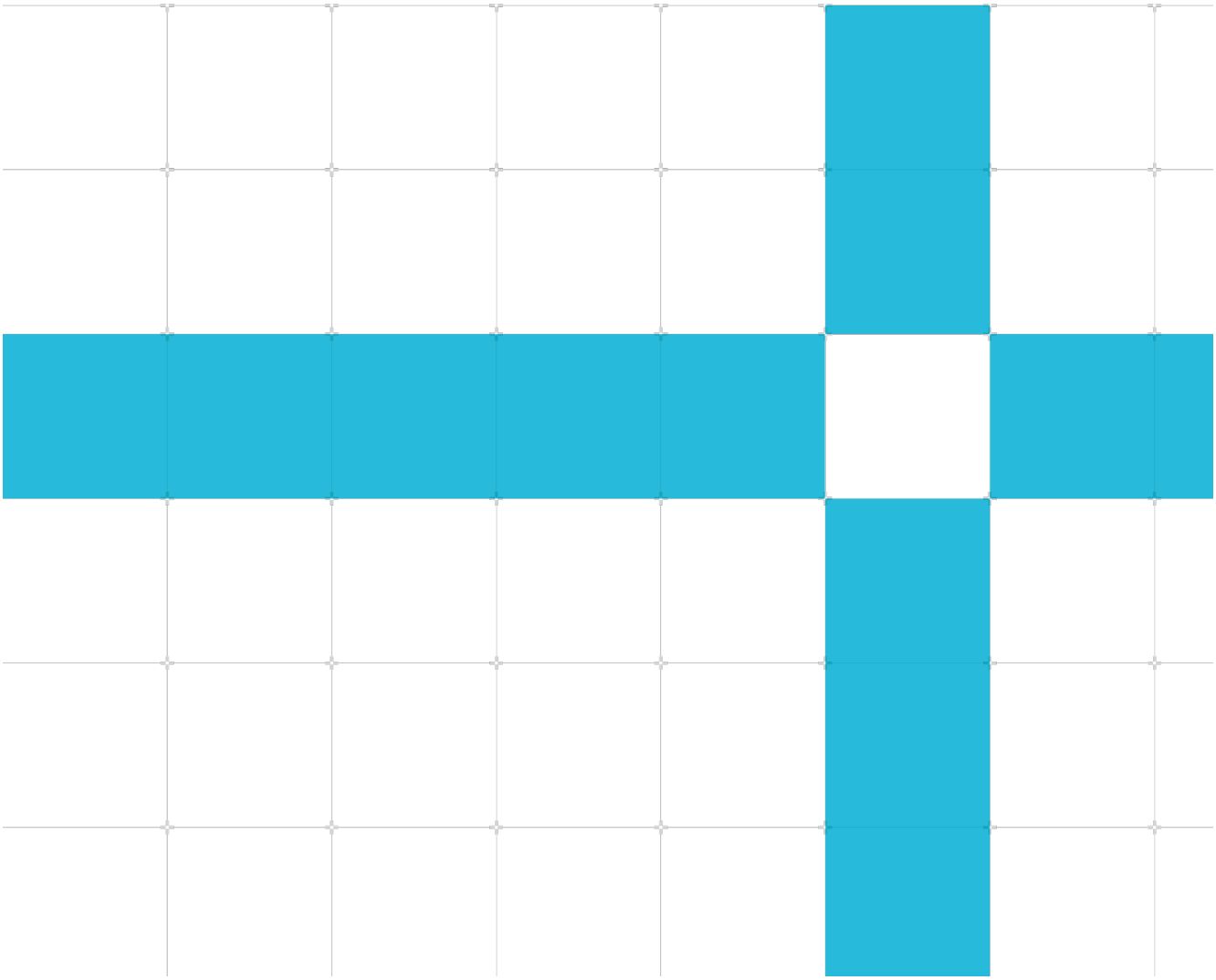
Date of issue: August 01, 2025

Non-Confidential

Document version: 8.0

Document ID: SDEN-1417924

This document contains all known errata since the r0p0 release of the product.



This document (SDEN_1417924_8.0_en) was issued on August 01, 2025.

There might be a later issue at **http://developer.arm.com/documentation/SDEN-1417924**

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.

If you find offensive language in this document, please email **terms@arm.com**.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on CoreLink MMU-600AE System Memory Management Unit, 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**.

Contents

Introduction

Scope

This document describes errata categorized by level of severity. Each description includes:

- The current status of the erratum.
- Where the implementation deviates from the specification and the conditions required for erroneous behavior to occur.
- The implications of the erratum with respect to typical applications.
- The application and limitations of a workaround where possible.

Categorization of errata

Errata are split into three levels of severity and further qualified as common or rare:

Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A (Rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B (Rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Change Control

Errata are listed in this section if they are new to the document, or marked as "updated" if there has been any change to the erratum text. Fixed errata are not shown as updated unless the erratum text has changed. The **errata summary table** identifies errata that have been fixed in each product revision.

August 01, 2025: Changes in document version v8.0

ID	Status	Area	Category	Summary
3533825	New	Programmer	Category B	Invalidation of nested translations currently in progress might not succeed
3533827	New	Programmer	Category C	Shareability incorrect for CMO in certain situations

July 18, 2025: Changes in document version v7.0

ID	Status	Area	Category	Summary
4014548	New	Programmer	Category B (rare)	Possible under-invalidation in TBU TLB when using STE CONT

July 03, 2024: Changes in document version v6.0

ID	Status	Area	Category	Summary
3605715	New	Programmer	Category B (rare)	ATS Invalidation request ATCI_PASID_GLOBAL does not specify SSID as required

November 18, 2022: Changes in document version v5.0

ID	Status	Area	Category	Summary
2662658	New	Programmer	Category B	FMU Ping counter issues when used with low-power interface.
2664509	New	Programmer	Category B	Clearing Error with coincident firing can cause failure to report by safety mechanism.
2748149	New	Programmer	Category C	Multiple Events reported to Event queue for a single transaction
2748152	New	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU

October 27, 2021: Changes in document version v4.0

ID	Status	Area	Category	Summary
2297885	New	Programmer	Category B (rare)	Global entries are not always invalidated as expected
2297884	New	Programmer	Category C	C_BAD_STREAMID not asserted in certain conditions

July 31, 2020: Changes in document version v3.0

No new or updated errata in this document version.

March 02, 2020: Changes in document version v2.0

ID	Status	Area	Category	Summary
1685319	New	Programmer	Category B (rare)	ATS requests can return global mappings
1500687	New	Programmer	Category C	FMU_SMINJERR write may lead to false error overflow indication in the error record
1685315	New	Programmer	Category C	Low performance for invalidation by IPA
1703639	New	Programmer	Category C	Incorrect ERR<n>STATUS.SERR value
1733426	New	Programmer	Category C	64 bit writes to FMU registers may not come into effect

March 15, 2019: Changes in document version v1.0

No errata in this document version.

Errata summary table

The errata associated with this product affect the product versions described in the following table.

ID	Area	Category	Summary	Found in versions	Fixed in version
2664509	Programmer	Category B	Clearing Error with coincident firing can cause failure to report by safety mechanism.	r0p0, r1p0	Open
2662658	Programmer	Category B	FMU Ping counter issues when used with low-power interface.	r0p0, r1p0	Open
3533825	Programmer	Category B	Invalidation of nested translations currently in progress might not succeed	r0p0	r1p0
2297885	Programmer	Category B (rare)	Global entries are not always invalidated as expected	r0p0, r1p0	Open
4014548	Programmer	Category B (rare)	Possible under-invalidation in TBU TLB when using STE CONT	r0p0, r1p0	Open
3605715	Programmer	Category B (rare)	ATS Invalidation request ATCI_PASID_GLOBAL does not specify SSID as required	r0p0, r1p0	Open
1685319	Programmer	Category B (rare)	ATS requests can return global mappings	r0p0	r1p0
2297884	Programmer	Category C	C_BAD_STREAMID not asserted in certain conditions	r0p0, r1p0	Open
2748149	Programmer	Category C	Multiple Events reported to Event queue for a single transaction	r0p0, r1p0	Open
2748152	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU	r0p0, r1p0	Open
1685315	Programmer	Category C	Low performance for invalidation by IPA	r0p0	r1p0
1703639	Programmer	Category C	Incorrect ERR<n>STATUS.SERR value	r0p0	r1p0
1500687	Programmer	Category C	FMU_SMINJERR write may lead to false error overflow indication in the error record	r0p0	r1p0
1733426	Programmer	Category C	64 bit writes to FMU registers may not come into effect	r0p0	r1p0
3533827	Programmer	Category C	Shareability incorrect for CMO in certain situations	r0p0	r1p0

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

2664509
Clearing Error with coincident firing can cause failure to report by safety mechanism

Status

Fault Type: Programmer CAT B
Fault Status: Present in r0p0,r1p0. Fixed in: Open

Description

When the TCU acknowledges the reporting of a fault by a remote TBU block, the TBU's safety mechanism (SM) could get into a state in which subsequent errors from the affected SM are not reported.

Conditions

For the issue to occur, the fault's ack/clear from the TCU(initiated by software) must arrive at the TBU's safety mechanism (SM) on the same cycle in which a subsequent new error is reported.

Configurations

All MMU-600AE FuSa configurations with TBUs. The TBUs contain SMs that detect errors and report them to the TCU. Errors detected and reported by TCU SMs are not affected.

Implications

If the condition occurs, subsequent errors might not be reported over the AXI-Stream interface until either:

- a clear is received (without an error being flagged by the SM)
- the SM is disabled and then re-enabled.

Workaround

There are two potential workarounds:
1) Reset the MMU when an error is detected. For this to work reliably, CEs should be treated as UEs by ensuring that the FMU_ERR<n>CTLR.CE_EN field is cleared.

2) When a remote MMU block reports an error (regardless if UE or CE), acknowledge and clear the error normally. After the error has been acknowledged, disable and reenable all relevant SMs belonging to the block reporting the error. Note: Enabling and Disabling only the SM reporting the error in the IER register is not considered to be a reliable workaround as the IER only captures the first SM reporting an error. Other SMs could have also reported an error on the same cycle as the clear.

2662658
FMU ping counter issues when used with low-power interface

Status

Fault Type: (GIC version Classified as Programmer CAT B)
Fault Status: Present in r0p0, r1p0. Fixed in: Open

Description

The MMU-600AE has a ping function which periodically pings the TBUs to ensure that the communications channels are still functional.

If TCU is requested to enter low-power mode (QSTOPPED) while the MMU is about to issue a ping message (but is otherwise idle), then it may incorrectly enter QSTOPPED.

Conditions

The erratum can occur if the following conditions are met at the same time:

1. The MMU is programmed to issue pings and it starts to issue a ping to any remote TBU.
2. The MMU is idle, apart from the ping logic.
3. A low-power entry request is received on the Q-Channel.

Configurations impacted

Any.

Implications

There are two implications:

- The ping interface will stop and not issue or report further pings until it is reprogrammed.
- The MMU may enter QSTOPPED with a ping message on an AXI-Stream bus. This may remain on the bus with the **VALID** signal HIGH and appear as multiple messages being issued on the bus.

Workaround

Do not use the ping functionality controlled by FMU registers FMU_PINGCTLR or PINGNOW if the LPI_CG interface is being used. The same functionality and fault coverage can be had by issuing an APB register read to each TBU and checking to see if the correct result is read. The recommended register to use for this purpose is read-only register SMMU_CIDR3. The result of the read should always be 0xB1.

3533825
Invalidation of nested translations currently in progress might not succeed

Status

Affects: MMU-600AE SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r1p0.

Description

After the MMU-600AE performs a Stage 2 or non-leaf Stage 1 Invalidation for translations in progress, some invalidated translations might still be used.

Conditions

This Erratum occurs when all the following conditions occur:

- The MMU-600AE is performing a page table walk for a transaction that requires stage 1 followed by stage 2 translation.
- As part of the page table walk, the MMU-600AE fetches a stage 1 page table descriptor and is performing the stage 2 translation to convert the descriptor to a physical address.
- The MMU-600AE receives a Stage 2 or a non-leaf Stage 1 invalidation, after the stage 1 non-leaf fetch completes, and before the corresponding stage 2 translation completes.
- For Stage 1 invalidation, the VA that is specified in the invalidation matches the *virtual address* (VA) descriptor undergoing stage 2 translation and does not match the VA of further descriptors. For stage 2 invalidation, the *Intermediate Physical Address* (IPA) that is specified in the invalidation matches the IPA of the stage 2 translation in progress.

When all the above conditions occur, the MMU-600AE erroneously updates the stage 1 and stage 2 walk cache with invalidated descriptors, that is not expected to occur.

Configurations affected

All configurations

Implications

It might be possible for a device to continue to access memory locations that it should not be possible to access after the invalidate and sync operation has completed.

Workaround

The following workarounds exist:

- Software can ensure that no concurrent invalidation of translations in progress occurs. Invalidation is issued only when the corresponding translations are complete.
- Software can repeat the combination of affected invalidation and SYNC commands once to ensure that invalidation takes effect.

Category B (rare)

2297885
Global entries are not always invalidated as expected

Status

- Fault Type: Programmer, CAT B (rare).
- Fault Status: Present in : r0p0, r1p0, Fixed in: Open

Description

A transaction whose translation walk is in progress (which is marked as a global translation and is subject to invalidation by TLBI_S_EL1_VA, TLBI_NS_EL1_VA, and TLBI_NS_EL2_VA operations) is not invalidated as expected, unless the ASID that is specified also matches.

Conditions

The erratum occurs when all the following conditions are true:

- A translation request is being translated that marks the translation as a global entry
- Stage 1 is enabled for the translation that is in progress
- The transaction is waiting for the translation result, and the translation in progress is currently either performing a TCU walk cache lookup or page table walk
- The SMMU receives one of TLBI_S_EL1_VA, TLBI_NS_EL1_VA, or TLBI_NS_EL2_VA operations
- The ASID for the translation that is in progress does not match the ASID that is specified in the invalidation command

When the above conditions occur, the transaction is not subjected to invalidation, and the translated entry is cached in the TLBs.

Configurations affected

All configurations are affected.

Implications

Because the intended invalidation operation does not succeed, the incorrect address is accessed post invalidation, and data corruption can occur. This issue does not cause a security violation because non-secure transactions cannot access Secure locations.

*Note: * Arm does not expect systems that are running Linux to be affected because up to at least v5.13, Linux does not use global translations.

Software workarounds

You can work around this issue by replacing the affected TLBI_***VA command with the corresponding TLBI***_VAA command.

4014548
Possible under-invalidation in TBU TLB when using STE CONT

Status

Affects: MMU-600AE SMMU -System Memory Mgmt Unit.
Fault Type: Programmer Category B (rare).
Fault Status: Present in: r0p0, r1p0 Fixed in: Open.

Description

When using STE.CONT != 0 and invalidation is performed using individual CMD_CFGI_STE commands instead of an appropriately sized CMD_CFGI_STE_RANGE command, it is possible for invalidated entries to remain present in the TBU TLB after a SYNC has completed.

Configurations affected

All configurations are affected.

Conditions

1. A set of contiguous STEs is cached in the TCU Configuration Cache, which was originally added to the Configuration Cache by a request to SID A.
2. Invalidations begin for the contiguous STEs by invalidating each STE in the contiguous region individually using CMD_CFGI_STE commands.
3. Before the invalidation for STE A is received, but after the invalidation for STE B has been received and applied to the TBU, the TBU receives a translation request for SID B.
4. The translation request for SID B hits in the Configuration Cache, because of the CONT bit, and updates the TLB in the TBU.
5. The invalidation occurs for SID A, which removes the contiguous entry from the TCU Configuration cache.
6. The invalidation sequence completes and a SYNC is performed.
7. A later request incorrectly hits on B in the TBU TLB.

Implications

STE.CONT is not commonly used and Linux does not support it. Where it is used, it is expected that it is invalidated using a CMD_CFGI_STE_RANGE command. However, in cases where it is used and CONT STEs are invalidated without using a single CMD_CFGI_STE_RANGE, under-invalidation occurs, with stale configuration data remaining present in the TBU TLB.

Workarounds

Any of the following workarounds prevent this erratum from occurring:

- Do not use STE.CONT != 0. Note: Linux does not use STE.CONT != 0.
- Ensure that any STE with CONT != 0 uses a single, appropriately sized, CMD_CFGI_STE_RANGE command instead of using individual CMD_CFGI_STE commands.
- If using a sequence of CMD_CFGI_STE commands, ensure the complete sequence is issued twice on each STE change.

3605715
ATS Invalidation request ATCI_PASID_GLOBAL does not specify SSID as required

Status

Affects: MMU-600AE SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Cat B Rare
Fault Status: Present in: r0p0, r1p0 Fixed in: Open.

Description

When the TCU raises an ATS Invalidation request ATCI_PASID_GLOBAL, the SSID field is required to be generated for the invalidation to be applied by PCIe Endpoint correctly. DTI Edition 3 specifies that the SSID field is invalid for ATCI_PASID_GLOBAL, so MMU-600AE always generates a value of 0, which results in the Global invalidation not being applied correctly by PCIe endpoints, if they do not support global invalidation.

Conditions

1. Software generates CMD_ATC_INV with SSV and G bits set as 1, indicating Global invalidation.
2. This results in TCU generating a ATCI_PASID_GLOBAL invalidation request on the DTI-ATS Interface.
3. The endpoint receiving the ATCI_PASID_GLOBAL does not support Global invalidation, so is expected by software to invalidate with the SubstreamID supplied, which has the value of 0.

When the above conditions occur, endpoint invalidation does not complete as expected by software.

Implications

When the above conditions occur, a stale translation can be used by the PCIe Endpoint, and that might lead to a virtualization hole and data corruption.

*Note: *Because ATS Translation requests are always Non-secure, Secure memory cannot be accessed or corrupted by a Non-secure access.
Note: Typical usage models of ATS do not use global mappings, and global invalidations are currently not supported by Linux, and therefore it is not likely to be an issue.

Workaround

You can apply the following software workaround:

Generate a non-global invalidation instead of a global invalidation. Because the invalidation is expected to be applied to a specific SubstreamID only, and because MMU-600AE does not generate Global translation, this does not result in any functional or performance implications.

1685319
ATS requests can return global mappings

Status

Fault Type: Programmer CAT B (rare)
Fault Status: Present in r0p0, Fixed in r1p0

Description

When the TCU returns an ATS Translation Completion for a request, the Global bit of the Translation Completion Data Entry must be returned zero in certain conditions, but instead is returned as 1.

Conditions

1. The ATS Translation request has the SubstreamID Valid bit set. DTI_TBU_TRANS_REQ.SSV = 1.
2. The Stream that is pointed to by the translation request requires Stage 1 translation. STE.Config is `3'b101` or STE.Config is `3'b111`.
3. The leaf entry for the Stage 1 page table has the nG (non-Global) bit set to 0, indicating a global page table entry.
4. The page table walk results in a successful ATS Translation response.

When the above conditions occur, the ATS Translation response has the Global bit incorrectly set to 1.

Implications

The above listed erratum has a functional implication if all of the following conditions are met:

1. ATS Translation requester uses the ATS Translation response with global bit set, for a different transaction with a different SubstreamID, but with the same Virtual Address.
2. The context descriptor pointed to by that different SubstreamID has a different ASET value from the one that generated the translation response.

When the above conditions occur, then an incorrect translation response can be used by the ATS Translation requester, that could lead to a virtualization hole and data corruption.
Because ATS Translation requests are always non-secure, there is no security violation because of this erratum.

Note: Typical usage models of ATS do not use global mappings, and is not currently supported by Linux, and therefore it is not likely to be an issue.

Workaround

You can apply the following workarounds:

Hardware workaround

1. The ATS Translation requester can ignore the Global bit returned in the ATS Translation response without any functional or performance implications and can always be considered to be 0.

Software workarounds

You can use one of the following software workarounds:

1. All the context descriptors for a given ATS StreamID must have a consistent ASET value.
2. All stage 1 page tables that ATS requests point to have no global mappings.

Category C

2297884
C_BAD_STREAMID not asserted in certain conditions

Status

Fault Type: Programmer, CAT C.
Fault Status: Present in: r0p0, r1p0, Fixed in: Open

Description

The MMU-600 does not generate C_BAD_STREAMID under certain conditions.

Conditions

This issue occurs when all the following conditions occur:

1. The value of STE.CONT is greater than SMMU_(S)_STRTAB_BASE_CFG.LOG2SIZE

1. A transaction is generated that uses this Stream table, and the resultant translation is cached in either the TBU uTLB, or MTLB, or both

1. Another transaction uses a Stream ID that is outside the range of SMMU_(S)_STRTAB_BASE_CFG.LOG2SIZE, but is within the range of Stream IDs that STE.CONT determines

When the above conditions occur, the TCU does not generate a C_BAD_STREAMID as expected.

Configurations affected

All configurations.

Implications

It is not expected that software programs an STE.CONT value that exceeds the span of the Stream table. It is also not expected that the device that is connected to the TBU generates an incorrect Stream ID.

If both these conditions occur, the transaction completes as determined by the Stream table information that is cached. This can result in data corruption or data being read by an unauthorized device.

Note: This erratum does not result in a Secure memory location being corrupted or read by a non-secure device.

Workaround

Ensure that the value of STE.CONT is not greater than SMMU_(S)_STRTAB_BASE_CFG.LOG2SIZE. This ensures that this erratum does not occur.

2748149
Multiple Events reported to Event queue for a single transaction

Status

Fault Type: Programmer, Cat C.
Fault Status: Present in: r0p0, r1p0. Fixed in: Open

Description

To avoid a synchronization deadlock, when a synchronization occurs, invalidated translations in the TBU might discard their translation and fetch a translation again.

Affected versions treat translation faults as invalidated. As a result, a transaction that faults in the TCU reports an event, and returns a fault response to the TBU might have its fault response discarded by the TBU and a new translation fetched from the TCU.

This new translation request can:

- Fault in the same manner, causing a repeated event to be reported
- Fault in a different manner, causing two different events to be reported for the same transaction
- Succeed, causing an event to be recorded for a transaction that has translated successfully

It is possible, but unlikely, that this sequence can occur multiple times.

Conditions

The issue can occur when:

- The event queue is enabled
- A transaction encounters a fault during translation that results in an event being reported
- That transaction has an ordering requirement in the TBU to be issued behind, or responded after, a second transaction
- A synchronization request arrives at the TBU from the TCU

Configurations affected

All configurations are affected.

Implications

The software that reads the content of the event queue might encounter events that relate to a transaction for which it has already read an event from the queue. The result of this depends on the behavior of the software.

Workaround

No workaround is required for the expected use models.

2748152
PMU event 0x2 (TLB Miss) triggers incorrectly in TCU

Status

Affects: MMU-600 SMMU - System Memory Mgmt
Fault Type: Programmer, Cat C.
Fault Status: Present in: r0p0, r1p0. Fixed in: Open

Description

PMU event 0x2 is expected to fire at most once per transaction when a TLB lookup result requires further memory access to produce a translation result. Affected versions only issue the PMU event when there is a complete miss in the Walk Cache on initial lookup. The event can trigger 0, 1 or 2 times for each time a transaction passes through the TCU and may trigger too many or too few times.

Conditions

This affects all Walk Cache hits that return a partial translation result when the PMU system is programmed to count the event.

Configurations affected

All configurations are affected.

Implications

A PMU counter counting transactions that require memory accesses to complete the translation may undercount or overcount them. This event is therefore not usable in practice.

Workaround

Depending on your translation regime, it may be possible to use the "S1L3 WC miss" event (`0x87`), or a different appropriate stage & level-specific miss event instead.

1685315
Low performance for invalidation by IPA

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r1p0

Description

When the MMU-600 receives an invalidation by Stage 2 VMID and IPA, the performance of the invalidation is lower than expected. The operation walks every entry of the TCU walk caches. The impact is higher when the cache size is larger.

Conditions

Either of the following occurs:

- The MMU-600 executes a CMD_TLBI_S2_IPA command through the Secure or Non-secure command queue.
- A processor executes a TLBI IPAS2{L}E1IS or TLBI IPAS2{L}E1OS instruction and the MMU-600 receives processor broadcast DVM operations.

Configurations affected

All configurations.

Implications

Arm expects Invalidate by IPA operations to be rare, so the impact is expected to be low.

If software invalidates large regions of IPA space using Invalidate by IPA operations instead of using a VMALL invalidate operation, then the performance might be noticeably impacted. For example, when a Virtual Machine is unmapped, software should use a VMALL invalidate operation.

There are no functional implications of this erratum and there is no impact on translation performance, or performance of any other invalidations.

Workaround

If required, you can use one of the following workarounds:

- Software should ensure that when large regions of IPA space are invalidated, a single VMALL

operation is used instead of invalidating each page individually.
- If it is practical to use larger pages, for example 2MB blocks instead of 4KB pages, then this significantly reduces the number of invalidate operations that are performed and improves performance.

1703639
Incorrect ERR<n>STATUS.SERR value

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r1p0

Description

Fault Management Unit (FMU) reports random hardware faults detected by asserting an interrupt. On receipt of an interrupt, the error recovery handling routine inspects the error records of the FMU to take appropriate actions. As part of this inspection the software performs the following steps

1. Read the ERRGSR register to find which error record number is reporting an error
2. Read ERR<n>STATUS register to get more information
   a. Bit[31] - V - would indicate this error record is in error
   b. Bits[25:24] - would indicate if this is a correctable error
   c. Bit[29] - would indicate if this is an uncorrectable error
   d. Bits[15:8] - identify the source of the error
   e. Bits[7:0] - SERR - provides additional information on architecturally defined primary code.
      - When there is no error this field return 8'd0
      - When there is an error this field should return a non-zero value.

The defect is that this field SERR always return 8'd0 even in case of error being reported where the reported value should be 8'd1.

Conditions

When error is reported, this field return incorrect value

Configurations affected

All configurations

Implications

Impact should be low as this is one of the fields which provides more additional information on the error. All the other information regarding the error is still valid (2.a to 2.d above). It may be confusing to software to read a error record indicating an error with this field set to 8'd0.

Workaround

Software should ignore this field. Use all other information described in points 2.a to 2.d above.

1500687
FMU_SMINJERR write may lead to false error overflow indication in the error record

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r1p0

Description

Doing software error injection using FMU_SMINJERR register may lead to a false error overflow indication in the error record

Condition

This errata occurs when all of the below are met

1. Software injects an fault using FMU_SMINJERR
2. The Fault Management Unit is clock gated due to inactivity on APB interface

Implication

Software fault injection is typically used to check whether the safety mechanism is operational and able to detect and report errors. Each software fault injected should be reported as one occurrence of error in the error record without an overflow. In this case, the safety mechanism does report an error but also reports an error overflow indication.

Software Workaround

During the software error injection sequence the clock gating to FMU should be disabled to avoid this errata. This can be achieved by writing to FMU_SMEN register the data 32'h1000_0000 [EN=0,BLK=0 (TCU) SMID=16 (fmu clock gate override)]

1733426
64 bit writes to FMU registers may not come into effect

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r1p0

Description

The Fault Management Unit (FMU) registers are protected using a lock and key mechanism. Once unlocked, the registers are locked again after a secure write to any register. Some FMU register are 64 bit registers wide, however the APB bus interface is 32 bits wide. The upper 32 bits of these registers is RESERVED or have read-only fields.

The design currently expects the software to perform 32 bit access to these 64 bit registers so that a KEY write can unlock the register before it updates that register.

If software makes a 64bit write to a register with data[63:0], the access will be split by the APB interconnect into two APB write of 32 bits each and will be forwarded to the FMU APB interface.

If these 32 bit writes reach the FMU interface in order "data[31:0]" followed by "data[63:32] " then lower 32 bits of the register (which contains meaningful contents) gets updated correctly. The upper 32 bits of the register would not be updated as the KEY would lock . This is not an issue as the upper 32 bits is RESERVED or read-only.

However, if the writes reaches FMU interface in order "data[63:32]" followed by "data[31:0]", then the lower 32 bits of the register would not be updated.

Configurations affected

All configurations

Conditions

All the following must be true

1. Software must perform 64 bit write
2. The APB interconnect must split the 64 bit write into two 32 bit access with the order "data[63:32]" followed by "data[31:0]"

Implication

The programming of FMU may not be successful.

Note: The FMU registers are mapped into "device memory". Typically, the APB interconnect would not reorder the 64 bit access in this specific order. For e.g. ARM NIC 400 APB interconnect always orders split writes as data[31:0] followed by data[63:32] in which case the system is not affected by errata.

Workaround

Hardware Workaround

The APB interconnect on receiving a 64 bit write with data[63:0] should split it into two APB writes in the order data[31:0] followed by data[63:32]

Software Workaround

The software should use 32 bit access to FMU registers.

3533827
Shareability incorrect for CMO in certain situations

Status

Affects: MMU-600AE SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r1p0.

Description

When the TBU translates a *Cache Maintenance Operation* (CMO), the **ARDOMAIN** value that the SMMU generates is incorrect in certain situations.

Conditions

1. A CleanShared, CleanInvalid, MakeInvalid, or CleanSharedPersist transaction is translated by the TBU.
2. The calculated cacheability of the transaction is Device or Non-cacheable.
3. The calculated shareability of the transaction is Non-shareable or Inner Shareable.
4. The interconnect downstream of the TBU distinguishes between Non-shareable, Inner Shareable, and Outer Shareable transactions.

The CMO is correctly translated with the cacheability on **ARCACHE** indicating Normal Write-back, because the cacheability is ignored for CMO transactions. This is to ensure that a master can clean caches after the translation tables have been updated to make the page Non-cacheable. However, the Arm architecture requires that a calculated cacheability of Device or Non-cacheable still overrides the shareability to Outer Shareable.

When all the above conditions are met, the transaction is output as Non-shareable or Inner Shareable whereas it should be output as Outer Shareable.

Implications

The CMO might not be propagated to all masters in the Outer Shareable shareability domain. However:

- Very few devices are capable of issuing CMO transactions.
- Most interconnects do not distinguish between Inner Shareable and Outer Shareable transactions.
- The requirement for the shareability to be upgraded in this case is for architectural consistency rather than known software usage models. Arm is not aware of any usage models that this behavior affects.

Workaround

No workaround is required.

Proprietary notice

(PRE-1121-V1.0)

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is for a product in development and is not final.

Product revision status

The rxpy identifier indicates the revision status of the product described in this manual, where:

rx

Identifies the major revision of the product.

py

Identifies the minor revision or modification status of the product.