CoreLink™ MMU-600 System Memory Management Unit

Software Developer Errata Notice

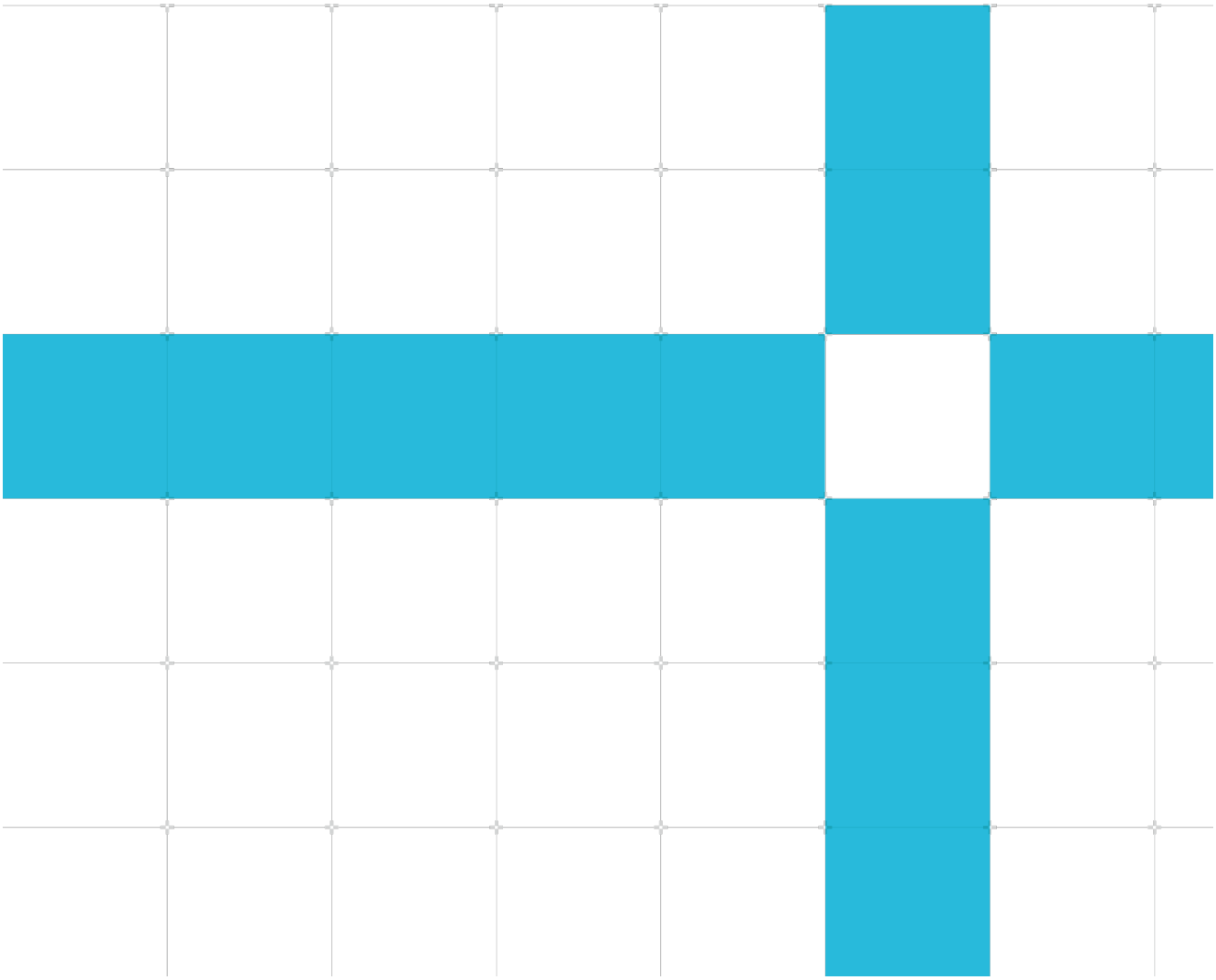
Date of issue: 15-Sep-2022

Non-Confidential

Document version: v9.0

Document ID: SDEN-946810

This document contains all known errata since the r0p1 release of the product.



Non-confidential proprietary notice

Arm Limited. Company 02557590 registered in England.

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on CoreLink™ MMU-600 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**.

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**.

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.

15-Sep-2022: Changes in document version v9.0

ID	Status	Area	Category	Summary
2666383	New	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU
2675030	New	Programmer	Category C	Multiple Events reported to Event queue for a single transaction

17-Sep-2021: Changes in document version v8.0

ID	Status	Area	Category	Summary
2214518	New	Programmer	Category B (rare)	Global entries are not always invalidated as expected
2121468	New	Programmer	Category C	C_BAD_STREAMID not asserted in certain conditions

07-May-2020: Changes in document version v7.0

No new or updated errata in this document version.

13-Dec-2019: Changes in document version v6.0

ID	Status	Area	Category	Summary
1365437	New	Programmer	Category B (rare)	ATS requests can return global mappings
1669310	New	Programmer	Category C	Low performance for invalidation by IPA

03-Oct-2018: Changes in document version v5.0

No new or updated errata in this document version.

30-Jul-2018: Changes in document version v4.0

ID	Status	Area	Category	Summary
1209401	New	Programmer	Category B	Invalidation of nested translations currently in progress might not succeed
1180733	New	Programmer	Category C	Shareability incorrect for CMO in certain situations

22-Mar-2018: Changes in document version v3.0

ID	Status	Area	Category	Summary
1076982	New	Programmer	Category B	SEV event not generated when command queue becomes non full

15-Dec-2017: Changes in document version v2.0

ID	Status	Area	Category	Summary
1025250	New	Programmer	Category B	Secure command queue blocked when an ATS Invalidation Completion timeout occurs and the Non-secure SMMU driver does not clear the resulting GERROR
1028311	New	Programmer	Category B	SMMU deadlocks if SMMUEN is cleared when command queue is processing CMD_ATC_INV command
998494	New	Programmer	Category C	Interrupt enable/disable not acknowledged when SMMU write is aborted
1001199	New	Programmer	Category C	Some PMCG performance monitoring events are inaccurate
1014060	New	Programmer	Category C	Incorrect shadow value registers on overflow capture
1041653	New	Programmer	Category C	ATS invalidation completion timeout not reported in presence of another GERROR

17-Aug-2017: 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
1209401	Programmer	Category B	Invalidation of nested translations currently in progress might not succeed	r0p1, r0p2, r1p0	r2p0
1028311	Programmer	Category B	SMMU deadlocks if SMMUEN is cleared when command queue is processing CMD_ATC_INV command	r0p1	r0p2
1025250	Programmer	Category B	Secure command queue blocked when an ATS Invalidation Completion timeout occurs and the Non-secure SMMU driver does not clear the resulting GERROR	r0p1	r0p2
1076982	Programmer	Category B	SEV event not generated when command queue becomes non full	r0p1, r0p2	r1p0
2214518	Programmer	Category B (rare)	Global entries are not always invalidated as expected	r0p1, r0p2, r1p0, r2p0, r2p1, r2p2	Open
1365437	Programmer	Category B (rare)	ATS requests can return global mappings	r0p1, r0p2, r1p0, r2p0, r2p1	r2p2
2121468	Programmer	Category C	C_BAD_STREAMID not asserted in certain conditions	r0p1, r0p2, r1p0, r2p0, r2p1, r2p2	Open
1669310	Programmer	Category C	Low performance for invalidation by IPA	r0p1, r0p2, r1p0, r2p0, r2p1	r2p2
1180733	Programmer	Category C	Shareability incorrect for CMO in certain situations	r0p1, r0p2, r1p0	r2p0
1001199	Programmer	Category C	Some PMCG performance monitoring events are inaccurate	r0p1	r0p2
1041653	Programmer	Category C	ATS invalidation completion timeout not reported in presence of another GERROR	r0p1, r0p2	r1p0
998494	Programmer	Category C	Interrupt enable/disable not acknowledged when SMMU write is aborted	r0p1	r0p2
1014060	Programmer	Category C	Incorrect shadow value registers on overflow capture	r0p1	r0p2

ID	Area	Category	Summary	Found in versions	Fixed in version
2675030	Programmer	Category C	Multiple Events reported to Event queue for a single transaction	r0p0, r0p1, r0p2, r1p0, r2p0, r2p1, r2p2	Open
2666383	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU	r0p0, r0p1, r0p2, r1p0, r2p0, r2p1, r2p2	Open

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

1209401
Invalidation of nested translations currently in progress might not succeed

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category B
Fault Status: Present in: r0p1, r0p2, r1p0. Fixed in: r2p0.

Description

After the MMU-600 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-600 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-600 fetches a stage 1 page table descriptor and is performing the stage 2 translation to convert the descriptor to a physical address.
- The MMU-600 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-600 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.

1028311
SMMU deadlocks if SMMUEN is cleared when command queue is processing CMD_ATC_INV command

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category B
Fault Status: Present in: r0p1. Fixed in: r0p2.

Description

When the SMMU executes a CMD_ATC_INV command from the Non-secure command queue at the same time as the Non-secure SMMU is disabled by writing 0 to SMMU_CR0.SMMUEN, then both the Non-secure and Secure command queues are permanently deadlocked, and the SMMU does not update SMMU_CR0ACK.SMMUEN to indicate that the SMMU has been disabled.

Conditions

The erratum occurs if the following sequence occurs:

1. A CMD_ATS_INV is read from the Non-secure command queue, but is not yet output on the TCU DTI interface.
2. On a specific clock cycle in between these two events, the Non-secure SMMU driver disables the Non-secure SMMU by writing SMMU_CR0.SMMUEN to LOW.

Implications

This erratum does not violate system security.

When this erratum occurs, the SMMU is unusable until it is reset. Translation continues as normal, but without a functioning command queue, the SMMU driver cannot function.

It is not expected that software disables the SMMU frequently, especially when an ATC invalidation command might be in progress, but if affected revisions of MMU-600 are supported, the SMMU driver should ensure that it implements the workaround to avoid the possibility of deadlock.

Workaround

Before the Non-secure SMMU driver clears SMMU_CR0.SMMUEN, it must ensure that either:

- The Non-secure command queue is empty.
- The Non-secure command queue is disabled by clearing SMMU_CR0.CMDQEN and waiting for SMMU_CR0ACK.CMDQEN to be cleared. **Note:** Clearing SMMU_CR0.SMMUEN and

SMMU_CR0.CMDQEN at the same time is not sufficient. SMMU_CR0.CMDQEN and SMMU_CR0ACK.CMDQEN must both be 0 before SMMU_CR0.SMMUEN is cleared.

1025250
Secure command queue blocked when an ATS Invalidation Completion timeout occurs and the Non-secure SMMU driver does not clear the resulting GERROR

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category B
Fault Status: Present in: r0p1. Fixed in: r0p2.

Description

The TCU does not increment the Secure command queue read index, when the read index is pointing at a CMD_SYNC command, and there is a pending ATS Invalidation timeout error.

Conditions

The erratum occurs if the following sequence occurs:

1. A CMD_ATS_INV is executed in the Non-secure command queue, that is passed to a PCIe endpoint.
2. The corresponding PCIe endpoint malfunctions and does not respond to the invalidation request.
3. A CMD_SYNC is executed in the Non-secure command queue, that is passed to the PCIe Root Complex.
4. The PCIe Root Complex waits for the endpoint to respond to the invalidation request, and eventually its timeout condition is reached.
5. The PCIe Root Complex acknowledges completion of the invalidation synchronization operation to the TCU, indicating an ATS Invalidation timeout by setting the DTI_ATS_SYNC_ACK.ERROR bit.
6. The TCU then raises a GERROR interrupt, toggles the state of SMMU_GERROR.CMDQ_ERR to indicate it is a command queue error, and sets SMMU_CMDQ_CONS.ERR=CERROR_ATC_INV_SYNC to indicate that it is caused by an ATS Invalidation timeout. At this point, the Non-secure command queue is stopped, with the SMMU_CMDQ_CONS.RD index pointing to the CMD_SYNC that caused the error.
7. The Non-secure SMMU driver does not clear the command queue error, leaving the Non-secure command queue stopped.
8. The Secure command queue contains a CMD_SYNC command.

When all the above conditions occur, the Secure CMD_SYNC is executed, but the SMMU_S_CMDQ_CONS.RD index is not updated, and the SMMU does not signal to software that the Secure CMD_SYNC is complete. The Secure CMD_SYNC is re-executed automatically until the Non-secure command queue is restarted by clearing the command queue error and a Non-secure CMD_SYNC completes successfully.

When the Non-secure SMMU driver encounters an ATS Invalidation timeout error, it normally raises an error to other software and restarts the command queue immediately, which causes the CMD_SYNC to be re-executed and clears the conditions necessary for the erratum. This does not result in an extra ATS Invalidation timeout error because no new CMD_ATS_INV command has been executed.

Implications

This erratum does not violate system security.

If the Non-secure SMMU driver is behaving normally, and it can operate independently of Secure software, then the effects of this erratum are not observed. However, if Secure software can stop Non-secure software while it is waiting for a Secure CMD_SYNC to complete, then the Non-secure SMMU driver might be unable to restart the command queue, exposing a race condition that could cause deadlock.

If a PCIe endpoint malfunctions and the Non-secure SMMU driver is unresponsive or malicious, then Secure software might not be able to use the SMMU Secure command queue. This represents a denial of service attack from Non-secure software to Secure software.

It is expected that Secure software implements a timeout to detect when the SMMU is non-responsive, and this erratum causes that timeout to be triggered.

Note: The SMMU only executes one command at a time from either command queue. If a PCIe endpoint does not respond to an ATC invalidate command, this can cause both command queues to be blocked until the Root Complex abandons the ATC invalidate operation and returns an ATC Invalidate timeout error with the following invalidation synchronization operation. This behavior is by design and is not affected by this erratum. The PCIe specification permits an endpoint to take up to 90 seconds to respond to an ATC invalidate operation, so a malfunctioning endpoint might cause the Secure SMMU timeout to be triggered, even when this erratum does not occur.

DVM invalidate operations are unaffected by this erratum and continue to be completed even when there are PCIe ATS invalidations pending.

Workaround

If the Secure software blocks Non-secure software while waiting for SMMU operations to complete, the Secure software can implement the following workaround to avoid the possibility of system deadlock:

1. If a Secure CMD_SYNC times out, inspect SMMU registers to check whether the following condition is met:
   a. SMMU_GERROR.CMDQ_ERR == !SMMU_GERRORN.CMDQ_ERR
   b. SMMU_CMDQ_CONS.ERR == CERROR_ATC_INV_SYNC
2. If this condition is met, return control to the Non-secure world for a limited amount of time to enable it to clear the error.
3. When the Secure software takes control again, the condition is still met:
   a. Ensure that the instruction that SMMU_CMDQ_CONS.RD points to is a CMD_SYNC instruction.
   b. Clear the error directly it by writing SMMU_GERRORN.CMDQ_ERR with the value of SMMU_GERROR.CMDQ_ERR.

c. The SMMU command queues should now be operational, but the Non-secure SMMU driver is not aware that the ATC invalidate operation failed, and is in an UNPREDICTABLE state. The Non-secure SMMU driver must therefore be restarted.

1076982
SEV event not generated when command queue becomes non full

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category B
Fault Status: Present in: r0p1, r0p2. Fixed in: r1p0.

Description

The SMMUv3 architecture requires that when SMMU_IDR0.SEV == 1, the SMMU triggers a WFE wake-up event when a Command queue becomes non-full and an agent external to the SMMU could have observed that the queue was previously full. The MMU-600 causes software that has executed a WFE to wake up by asserting the **evento** output.

In revisions of MMU-600 affected by this erratum, **evento** is not asserted when the Command queue transitions from full to non-full.

Conditions

Software fills the MMU-600 Command queue, then executes a WFE instruction.

Implications

The software does not restart when the Command queue becomes non-full, and instead restarts when the next WFE restart event occurs, for example the next timer interrupt.

Workaround

When the Command queue becomes full, instead of executing a WFE instruction, software can poll the SMMU_(S_)CMDQ_CONS register until it observes that the Command queue is not full.

This behavior is expected for SMMUv3 implementations where SMMU_IDR0.SEV == 0, so an alternative approach is to treat SMMU_IDR0.SEV as 0.

Category B (rare)

2214518
Global entries are not always invalidated as expected

Status

- Affects: MMU-600 SMMU - System Memory Mgmt.
- Fault Type: Programmer, Category B (rare).
- Fault Status: r0p1, r0p2, r1p0, r2p0, r2p1, r2p2. 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.

1365437
ATS requests can return global mappings

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category B (rare)
Fault Status: Present in: r0p1, r0p2, r1p0, r2p0, r2p1. Fixed in: r2p2.

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

2121468
C_BAD_STREAMID not asserted in certain conditions

Status

Affects: MMU-600 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C.
Fault Status: Present in: r0p1, r0p2, r1p0, r2p0, r2p1,r2p2. 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
2. A transaction is generated that uses this Stream table, and the resultant translation is cached in either the TBU uTLB, or MTLB, or both
3. 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.

1669310
Low performance for invalidation by IPA

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p1, r0p2, r1p0, r2p0, r2p1. Fixed in: r2p2.

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.

1180733
Shareability incorrect for CMO in certain situations

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p1, r0p2, r1p0. Fixed in: r2p0.

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.

1001199
Some PMCG performance monitoring events are inaccurate

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p1. Fixed in: r0p2.

Description

Some PMCG performance monitoring events are inaccurate.

Conditions

This erratum affects the following conditions:

- TCU event `0x03`, Configuration cache miss caused by transaction or translation request. When multiple translations miss in the configuration cache but can share a single configuration table walk, this event counts once per translation. The correct behavior it to count only once because a single configuration table walk occurs.
- TCU event `0x02`, TLB miss caused by incoming transaction or translation request. When a translation requires both stage 1 and stage 2 translation, and it misses in every walk cache stage, then this event does not count. This is rare because most translations hit in at least the level 0 or level 1 walk caches.
- TCU event `0x91`, Buffered translation. This event counts when the translation request buffer is not empty and new translations arrive, even if some translation slots are unused. The correct behavior is to only count when new translations are written to the translation request buffer because all translation slots are full.
- TBU event `0x89`, Write data uses write data buffer. This event might count unexpectedly when a write transaction that uses the write data buffer, and does not match the PMCG filter conditions, coincides with a read transaction that does match the PMCG filter conditions.

Implications

The counter events are still usable, even though they are slightly incorrect.

Workaround

No workaround is necessary.

1041653
ATS invalidation completion timeout not reported in presence of another GERROR

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p1, r0p2. Fixed in: r1p0.

Description

If an ATS Invalidation Completion timeout error occurs at the same time as a TCU event queue, PRI queue, or an MSI transaction results in an abort, then the ATS Invalidation Completion timeout error might not be reported to software.

This can only occur when the following are all true:

- A PCIe endpoint malfunctions, causing an ATS Invalidation Completion timeout to occur.
- The MMU-600 or the system is incorrectly configured, causing the memory system to abort a TCU queue or MSI write transaction.
- Specific timing conditions are met inside the MMU-600.

Conditions

The erratum occurs if the following sequence occurs:

1. A CMD_ATS_INV is executed in the Non-secure command queue, and it is passed to a PCIe endpoint.
2. The corresponding PCIe endpoint malfunctions and does not respond to the invalidation request.
3. A CMD_SYNC in executed in the Non-secure command queue, and it is passed to the PCIe Root Complex.
4. The PCIe Root Complex waits for the endpoint to respond to the invalidation request, and eventually its timeout condition is reached.
5. The PCIe Root Complex acknowledges completion of the invalidation synchronization operation to the TCU, indicating an ATS Invalidation timeout by setting the DTI_ATS_SYNC_ACK.ERROR bit, causing a command queue global error.
6. On a specific cycle at this point, another global error occurs, that is not a command queue error. This can occur when either an event queue write, PRI queue write, or MSI write is terminated with an abort.

When the above conditions all occur, the TCU does not set the SMMU_GERROR.CMDQ_ERR bit for this error. However, it does correctly set the SMMU_CMDQ_CONS.ERR register to the value CERROR_ATC_INV_SYNC.

Because no command queue error is raised, execution of commands in the command queue restarts immediately, causing the CMD_SYNC that caused the error to be retried. This is completed without an error, because the new CMD_SYNC is executed before any new ATC Invalidation commands.

Implications

When the system or MMU-600 is incorrectly configured and an endpoint malfunctions, software might consider ATC invalidation commands to have completed successfully that have not been completed.

Secure software is unaffected because Secure software does not issue ATC Invalidate operations.

Workaround

In correctly configured systems, this erratum does not occur. However, some software must work around this erratum.

This workaround must only be used with the MMU-600 because it takes advantage of a property of the MMU-600 that the SMMUv3 does not require. In the MMU-600, when SMMU_CMDQ_CONS.ERR is set to a value, that value is maintained until the next command queue error.
A CMD_SYNC is completed successfully when the SMMU_CMDQ_CONS.RD index passes the location of the CMD_SYNC, and the SMMU_CMDQ_CONS.ERR field is zero.
Whenever SMMU_CMDQ_CONS is read, the ERR field must be read. If ERR is not zero, then software must assume that an ATS Invalidation Completion timeout error has occurred. When this occurs, software must perform the following:

1. Handle and clear any global errors reported in SMMU_GERROR.
2. Re-issue to the command queue previous ATC Invalidate commands that have not been followed by a successfully executed CMD_SYNC.
3. Issue a new CMD_SYNC to the command queue.
4. Wait for the command queue to be empty.
5. Disable the command queue.
6. Clear SMMU_CMDQ_CONS.ERR.
7. Re-enable the command queue.

998494
Interrupt enable/disable not acknowledged when SMMU write is aborted

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p1. Fixed in: r0p2.

Description

When software enables or disables interrupts, the update might not be acknowledged if it coincides with an abort response from a write that the TCU made, that could cause a software deadlock.

Conditions

- The TCU issues an Event Queue write or MSI write transaction.
- Software writes to the SMMU_(S_)IRQ_CTRL register. The ACK for this is waiting for the write response to be received.
- Another Event Queue write or MSI Write transaction must be scheduled by the TCU because one of the following conditions occurs:
  - The system returns a write abort for the TCU write.
  - Another MSI write transaction is scheduled because of SYNC completion on the Command Queue, or a Command Queue Error.
  - The TCU issues an Event Queue write.
- Software waits for the TCU to update SMMU_(S_)IRQ_CTRLACK to match the new value of SMMU_(S_)IRQ_CTRL.

Configurations affected

All configurations.

Implications

If this erratum occurs, then software might enter an infinite loop, waiting for the value of SMMU_(S_)IRQ_CTRLACK to change.

This erratum can occur in one of the following scenarios:

- The software that controls the SMMU programs the event queue or IRQ registers incorrectly, so that the system returns an abort when the TCU issues an event queue write or MSI write. This does not occur if the SMMU software is correct.
- A random system error occurs that causes a TCU write to return an error response. This is expected to be rare, because random system errors are rare, and updates to the SMMU_(S_)IRQ_CTRL register

are rare.

Workarounds

For most systems, it is not necessary to work around this erratum.

Software can work around this erratum by implementing one of the following:

1. Modify the SMMU_IRQ_CTRL register only when the Event Queue is disabled and MSIs are disabled.
2. Implement a timeout in the SMMU_(S_)IRQ_CTRLACK polling loop and retry updating SMMU_(S_)IRQ_CTRL if the timeout occurs.

1014060
Incorrect shadow value registers on overflow capture

Status

Affects: MMU-600 SMMU -System Memory Mgmt Unit
Fault Type: Programmer, Category C
Fault Status: Present in: r0p1. Fixed in: r0p2.

Description

When a counter overflow occurs, all event counters are captured on the same clock as the overflow. If another event occurs in the same clock as the overflow, this could mean that the event shadow registers show one less count than the expected count. Also, the overflowed event shadow register captures a value of `0xFFFF_FFFF` instead of `0x0`.

Conditions

This event occurs when a counter overflows, and other events that are enabled to be counted occur in the same clock as the overflow.

Configurations affected

All configurations.

Implications

If software uses overflow capture as a mechanism to check the status of other events, then the events could appear as one less than the actual number of events. Because there is no specific significance to the count, this might not make any difference to the usability of event counts.

Workarounds

No workaround is required.

2675030
Multiple Events reported to Event queue for a single transaction

Status

Fault Type: Programmer, Cat C.
Fault Status: Present in: r0p0, r0p1, r0p2, r1p0, r2p0, r2p1, r2p2. 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.

2666383
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, r0p1, r0p2, r1p0, r2p0, r2p1, r2p2 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.