MMU-700 System Memory Management Unit

Software Developer Errata Notice

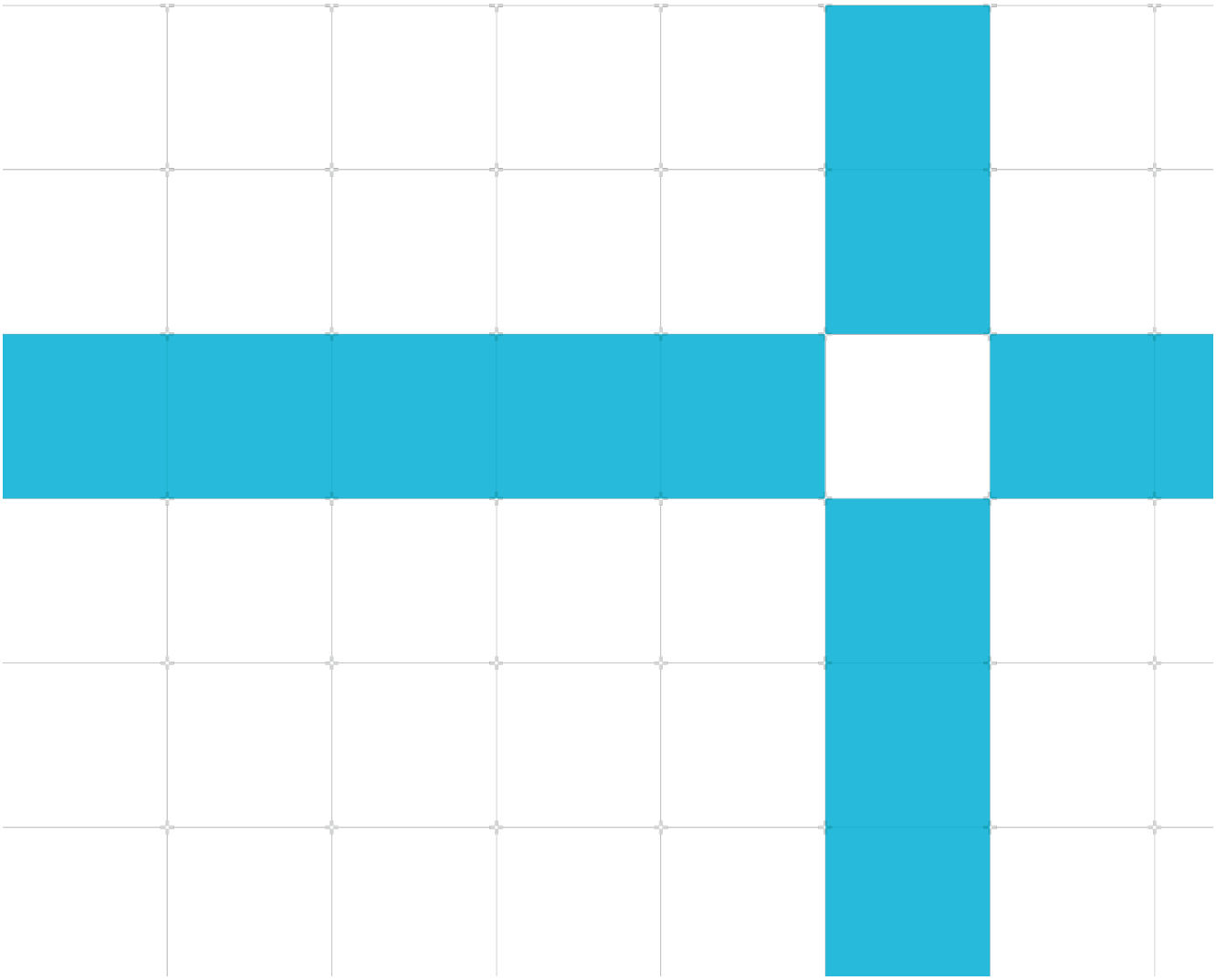
Date of issue: 09-Jun-2023

Non-Confidential

Document version: 11.0

Document ID: SDEN-1786925

This document contains all known errata since the r0p0 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.

Product status

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

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on MMU-700 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.

09-Jun-2023: Changes in document version v11.0

ID	Status	Area	Category	Summary
2133013	Updated	Programmer	Category B	Stage 1 DVM invalidation operations of translations using TTB1 might not succeed
2239241	Updated	Programmer	Category B	MPAM CMAX register write uses incorrect bits from the write data
2428613	Updated	Programmer	Category B	TBU or ATS connection process can sometimes cause deadlock
2650791	Updated	Programmer	Category B	MPAM monitor values incorrect if configuration changed while enabled
2808817	Updated	Programmer	Category B (rare)	A device that can issue FIXED bursts can bypass hypervisor Force Write-Back controls to read old memory contents
2288256	Updated	Programmer	Category C	Configuration and translation structures might be accessed by prefetch transactions while SMMU(_S)_CR0.SMMUEN=0 and SMMU(_S)_CR0ACK.SMMUEN=0
2621157	Updated	Programmer	Category C	Multiple Events reported to Event queue for a single transaction
2626480	Updated	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU
2674701	Updated	Programmer	Category C	Read of TCU_MSMON_CAPT_EVNT register returns data instead of zero
2803259	Updated	Programmer	Category C	Incorrect merging of S1 Allocation hints
2900780	Updated	Programmer	Category C	Stream table base address not properly aligned when log2size is configured greater than stream ID size
2945011	New	Programmer	Category C	Address size fault raised when some RES0 page table descriptor bits have non-zero values

19-Apr-2023: Changes in document version v10.0

ID	Status	Area	Category	Summary
2216408	Updated	Programmer	Category B	Invalidated transactions might sometimes complete after SYNC completion
2268618	Updated	Programmer	Category B	TLBI_(R)IPAS2* and CMD_TLBI(_S)_S2_IPA invalidation of nested translations currently in progress might not succeed
2428613	Updated	Programmer	Category B	TBU or ATS connection process can sometimes cause deadlock
2812531	Updated	Programmer	Category B	Walk cache updates sent incorrectly after invalidation
2214515	Updated	Programmer	Category B (rare)	Global entries are not always invalidated as expected
2900780	New	Programmer	Category C	Stream table base address not properly aligned when log2size is configured greater than stream ID size

17-Mar-2023: Changes in document version v9.0

ID	Status	Area	Category	Summary
2812531	New	Programmer	Category B	Walk cache updates sent incorrectly after invalidation
2803259	New	Programmer	Category C	Incorrect merging of S1 Allocation hints

21-Dec-2022: Changes in document version v8.0

ID	Status	Area	Category	Summary
2133013	Updated	Programmer	Category B	Stage 1 DVM invalidation operations of translations using TTB1 might not succeed
2808817	New	Programmer	Category B (rare)	A device that can issue FIXED bursts can bypass hypervisor Force Write-Back controls to read old memory contents

18-Jul-2022: Changes in document version v7.0

ID	Status	Area	Category	Summary
2650791	New	Programmer	Category B	MPAM monitor values incorrect if configuration changed while enabled
2621157	New	Programmer	Category C	Multiple Events reported to Event queue for a single transaction
2674701	New	Programmer	Category C	Read of TCU_MSMON_CAPT_EVNT register returns data instead of zero

26-May-2022: Changes in document version v6.0

ID	Status	Area	Category	Summary
2428613	New	Programmer	Category B	TBU or ATS connection process can sometimes cause deadlock
2626480	New	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU

17-Sep-2021: Changes in document version v5.0

ID	Status	Area	Category	Summary
2133013	Updated	Programmer	Category B	Stage 1 DVM invalidation operations of translations using TTB1 might not succeed
2216408	New	Programmer	Category B	Invalidated transactions might sometimes complete after SYNC completion
2239241	New	Programmer	Category B	MPAM CMAX register write uses incorrect bits from the write data
2268618	New	Programmer	Category B	TLBI_(R)IPAS2* and CMD_TLBI(_S)_S2_IPA invalidation of nested translations currently in progress might not succeed
2214515	New	Programmer	Category B (rare)	Global entries are not always invalidated as expected
2288256	New	Programmer	Category C	Configuration and translation structures might be accessed by prefetch transactions while SMMU(_S)_CR0.SMMUEN=0 and SMMU(_S)_CR0ACK.SMMUEN=0

24-May-2021: Changes in document version v4.0

ID	Status	Area	Category	Summary
2133013	New	Programmer	Category B	Stage 1 DVM invalidation operations of translations using TTB1 might not succeed

26-Mar-2021: Changes in document version v3.0

ID	Status	Area	Category	Summary
2105620	New	Programmer	Category B	TLB allocation inefficient in certain conditions
2093540	New	Programmer	Category C	TCU Performance Monitoring Event indicating Config Cache Read inaccurate
2120193	New	Programmer	Category C	C_BAD_STREAMID not asserted under certain conditions

30-Oct-2020: Changes in document version v2.0

ID	Status	Area	Category	Summary
1917448	New	Programmer	Category A	HTTU operations cannot be used when stage 1 followed by stage 2 translation is enabled
1781134	New	Programmer	Category B	MPAM partition hard limit sometimes exceeded
1832597	New	Programmer	Category B	Secure transactions result in incorrect C_BAD_STE event under certain conditions
1848465	New	Programmer	Category B	Termination of stalls might not occur sometimes
1854758	New	Programmer	Category B	Register reads to the TBU do not complete under certain conditions
1932231	New	Programmer	Category B	Incorrect read data for some registers
1941946	New	Programmer	Category B	Some invalidation operations might not succeed
1768143	New	Programmer	Category C	Non-zero value of SSID generated in PRI response when SSV is 0
1781603	Updated	Programmer	Category C	Event queue not overflowing when filled and fault not reported
1784630	Updated	Programmer	Category C	Incorrect IPA in fault syndrome reported for HTTU enabled nested translations
1871067	New	Programmer	Category C	C_BAD_STE not asserted under certain conditions
1877245	New	Programmer	Category C	RAS Error can result in RAS registers read data being corrupted
1879539	New	Programmer	Category C	Incorrect fault data sometimes reported
1912550	New	Programmer	Category C	Incorrect shareability for Cache Maintenance Operations (CMOs)

30-Mar-2020: Changes in document version v1.0

ID	Status	Area	Category	Summary
1781603	New	Programmer	Category C	Event queue not overflowing when filled and fault not reported
1784630	New	Programmer	Category C	Incorrect IPA in fault syndrome reported for HTTU enabled nested translations

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
1917448	Programmer	Category A	HTTU operations cannot be used when stage 1 followed by stage 2 translation is enabled	r0p0	r0p1
1781134	Programmer	Category B	MPAM partition hard limit sometimes exceeded	r0p0	r0p1
1832597	Programmer	Category B	Secure transactions result in incorrect C_BAD_STE event under certain conditions	r0p0	r0p1
1848465	Programmer	Category B	Termination of stalls might not occur sometimes	r0p0	r0p1
1854758	Programmer	Category B	Register reads to the TBU do not complete under certain conditions	r0p0	r0p1
1932231	Programmer	Category B	Incorrect read data for some registers	r0p0	r0p1
1941946	Programmer	Category B	Some invalidation operations might not succeed	r0p0	r0p1
2105620	Programmer	Category B	TLB allocation inefficient in certain conditions	r0p0, r0p1	r1p0
2133013	Programmer	Category B	Stage 1 DVM invalidation operations of translations using TTB1 might not succeed	r0p0, r0p1, r1p0, r1p1	r1p2
2216408	Programmer	Category B	Invalidated transactions might sometimes complete after SYNC completion	r0p0, r0p1, r1p0	r1p1
2239241	Programmer	Category B	MPAM CMAX register write uses incorrect bits from the write data	r0p0, r0p1, r1p0, r1p1	r1p2
2268618	Programmer	Category B	TLBI_(R)IPAS2* and CMD_TLBI(_S)_S2_IPA invalidation of nested translations currently in progress might not succeed	r0p0, r0p1, r1p0	r1p1
2428613	Programmer	Category B	TBU or ATS connection process can sometimes cause deadlock	r0p0, r0p1, r1p0, r1p1	r1p2
2650791	Programmer	Category B	MPAM monitor values incorrect if configuration changed while enabled	r0p0, r0p1, r1p0, r1p1	r1p2
2812531	Programmer	Category B	Walk cache updates sent incorrectly after invalidation	r0p0, r0p1, r1p0	r1p1

ID	Area	Category	Summary	Found in versions	Fixed in version
2214515	Programmer	Category B (rare)	Global entries are not always invalidated as expected	r0p0, r0p1, r1p0	r1p1
2808817	Programmer	Category B (rare)	A device that can issue FIXED bursts can bypass hypervisor Force Write-Back controls to read old memory contents	r0p0, r0p1, r1p0, r1p1	r1p2
1768143	Programmer	Category C	Non-zero value of SSID generated in PRI response when SSV is 0	r0p0	r0p1
1781603	Programmer	Category C	Event queue not overflowing when filled and fault not reported	r0p0	r0p1
1784630	Programmer	Category C	Incorrect IPA in fault syndrome reported for HTTU enabled nested translations	r0p0	r0p1
1871067	Programmer	Category C	C_BAD_STE not asserted under certain conditions	r0p0	r0p1
1877245	Programmer	Category C	RAS Error can result in RAS registers read data being corrupted	r0p0	r0p1
1879539	Programmer	Category C	Incorrect fault data sometimes reported	r0p0	r0p1
1912550	Programmer	Category C	Incorrect shareability for Cache Maintenance Operations (CMOs)	r0p0	r0p1
2093540	Programmer	Category C	TCU Performance Monitoring Event indicating Config Cache Read inaccurate	r0p0, r0p1	r1p0
2120193	Programmer	Category C	C_BAD_STREAMID not asserted under certain conditions	r0p0, r0p1	r1p0
2288256	Programmer	Category C	Configuration and translation structures might be accessed by prefetch transactions while SMMU(_S)_CR0.SMMUEN=0 and SMMU(_S)_CR0ACK.SMMUEN=0	r0p0, r0p1, r1p0, r1p1	r1p2
2621157	Programmer	Category C	Multiple Events reported to Event queue for a single transaction	r0p0, r0p1, r1p0, r1p1	r1p2
2626480	Programmer	Category C	PMU event 0x2 (TLB Miss) triggers incorrectly in TCU	r0p0, r0p1, r1p0, r1p1	r1p2
2674701	Programmer	Category C	Read of TCU_MSMON_CAPT_EVNT register returns data instead of zero	r0p0, r0p1, r1p0, r1p1	r1p2
2803259	Programmer	Category C	Incorrect merging of S1 Allocation hints	r0p0, r0p1, r1p0, r1p1	r1p2

ID	Area	Category	Summary	Found in versions	Fixed in version
2900780	Programmer	Category C	Stream table base address not properly aligned when log2size is configured greater than stream ID size	r0p0, r0p1, r1p0, r1p1	r1p2
2945011	Programmer	Category C	Address size fault raised when some RES0 page table descriptor bits have non-zero values	r0p0, r0p1, r1p0, r1p1, r1p2	Open

Errata descriptions

Category A

1917448
HTTU operations cannot be used when stage 1 and stage 2 translation is enabled

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category A
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

Hardware Translation Table Update (HTTU) operations can result in AXI Protocol violation in certain conditions.

Conditions

This issue occurs when all the following conditions occur:

- Two or more transactions require *Hardware Translation Table Update* (HTTU)
- One of the transactions (Transaction A) requires stage 1 followed by stage 2 translation, and has HTTU enabled for stage 2
- In the process of performing translation for Transaction A, a hardware translation table update for stage 2 translation is performed at least once, and is in the process of generating an update for another stage 2 translation
- Another transaction (Transaction B) has initiated an atomic transaction as part of HTTU and is waiting for write or read responses
- A write or read response is received for Transaction B at the same time as Transaction A is initiating its atomic transaction for the stage 2 translation update

Configurations affected

All configurations.

Implications

- When the above conditions occur, the TCU can generate two outstanding atomic transactions with

same AXI ID
- This is an AXI Protocol violation and could result in UNPREDICTABLE behavior in the downstream system

Workaround

No workaround.

Category A (rare)

There are no errata in this category.

Category B

1781134
MPAM partition hard limit sometimes exceeded

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

In certain conditions, the cache blocks, Main TLB, configuration cache, and walk cache exceed the programmed partition limits.

Conditions

This issue occurs when any of the following conditions occur:

- The **`MPAMCFG_CMAX`** value is changed dynamically to be lower than the current occupancy. **Note:** This does not prevent new allocations occurring to that partition.
- Global entries not following **`PARTID_WIDTH`** for the specific ASID

Configurations affected

All configurations.

Implications

- Global entries are not expected to be used in the SMMU, and if a few global entries are used, this does not result in a high impact on performance
- Incorrect allocation because of dynamic **`MPAMCFG_CMAX`** modification might result in some streams having reduced performance

Workaround

Perform an **`INV_ALL`** before modifying the **`MPAMFG_CMAX`** value to ensure correct allocation of cache resources.

1832597
Secure transactions result in incorrect C_BAD_STE event under certain conditions

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

Secure transactions result in the C_BAD_STE event being raised incorrectly under certain conditions.

Conditions

This issue occurs when all the following conditions occur:

- The MMU-700 receives a transaction with a SECSID value of 1, that requires stage 2 translation
- The stream table descriptor corresponding to that transaction has STE.s2s bit set to 1
- SMMU_CR0.NSSTALLD is set to 1

Under these conditions, an incorrect C_BAD_STE event is raised.

Configurations affected

All configurations.

Implications

This issue results in Secure transactions being unable to progress. Secure translations that require stage 2 translation and stall mode, when Non-secure stall is disabled, are expected to be rare.

Workaround

Not using stall mode for Secure transactions that require stage 2 translation ensures that this issue does not occur.

1848465
Termination of stalls might sometimes not occur

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

The **CMD_STALL_TERM** command does not terminate all transactions under certain conditions.

Conditions

The conditions are as follows:

- One or more transactions that the MMU-700 receives encounters a faulting condition during the translation process that is configured to stall
- The event queue corresponding to the security of the transactions is either not enabled or is full, resulting in failure to report the stall
- Software issues a **CMD_STALL_TERM** that requires the non-reported stalled transactions to be terminated

When all the above conditions occur, these non-reported stalled transactions are not aborted. Instead, they retry the translation process when the event queue is enabled or is not full.

Configurations affected

All configurations.

Implications

Transactions that are expected to have been completed are visible to software as restarted transactions. This issue does not cause any other security or functional issues.

Workaround

Ensure that the event queue is enabled and not full before a **CMD_STALL_TERM** is issued to avoid this issue.

1854758
Register reads to the TBU do not complete under certain conditions

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

Register reads to the TBU do not complete in certain conditions, leading to the APB port being locked up.

Conditions

This issue occurs when all the following conditions occur:

- The Command queue in the TCU is enabled, and one or more commands are pending to be processed by the TCU
- A register read is initiated on the APB bus to one of the TBU address regions
- The timing of the register read data coincides with the process of the consumer pointer being updated for the command queue

Configurations affected

All configurations.

Implications

When all the above conditions occur, the register read never completes and the APB bus is deadlocked. This does not affect register reads to the TCU address region.

Workaround

Ensure that no register accesses are pending to the TBU address region, when commands are pending to be processed in the command queue.

1932231
Incorrect read data for some registers

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

The following registers return incorrect data when read:

- TCU_CFG register read returns 0 for the **`XLATE_SLOTS`** field, instead of the configured **`TCUCFG_PTW_SLOTS`**
- The SMMU_MPAMIDR and SMMU_S_MPAMIDR register reads return a value of **`15'h000A`**, instead of the configured value
- The following MPAM registers return the value programmed beyond **`PARTID_WIDTH`**:
  - MPAMCFG_PART_SEL
  - MPAMCFG_CMAX
  - MSMON_CFG_MON_SEL
  - MSMON_CFG_CSU_FLT
  - MSMON_CFG_CSU_CTL
  - MSMON_CAPT_EVNT
- When **`MTLB_WIDTH`** is 0, Secure and Non-secure MPAMF_IDR registers indicate that MPAM functionality is still present
- TCU_NODE_STATUS does not reflect the PRI, it only reflects the ATS masters that are connected

Configurations affected

All configurations.

Implications

- Incorrect read data on microarchitectural registers is not expected to cause any problems to standard software
- Incorrect **`PARTID_WIDTH`** read on the MPAM capability registers can result in complete capability of the MPAM functionality not being taken advantage of
- Bits beyond **`PARTID_WIDTH`** that are programmed into the MPAM registers do not result in any functional or performance issues. Only the register data read back will have the values that are written into the bits beyond **`PARTID_WIDTH`**.

Workaround

- Software can use the TCU_SYSDISC11 and TBU_SYSDISC11 registers to correctly identify the **PARTID_WIDTH** capability of the TCU and TBU
- Software can use the TCU_SYSDISC5 register for the **TCUCFG_PTW_SLOTS** value

1941946
Some invalidation operations might not succeed

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category B
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

After the MMU-700 performs some invalidation operations, some invalidated translations might still be used.

Conditions

The MMU-700 performs an invalidation operation because of one of the following:

- A command in the command queue
- A TLB invalidate operation on the DVM interface
- An access to one of the following register bits, which are managed inside the MMU-700 using invalidation mechanisms:
  - SMMU_(S_)CR0.SMMUEN
  - SMMU_CR0.ATSCHK
  - SMMU_(S_)GBPA.*
  - SMMU_S_INIT.INV_ALL
  - SMMU_S_CR0.SIF

A transaction is using an invalidated translation at the same time as it is being invalidated.

Configurations affected

All configurations.

Implications

A device might continue to be able to access memory locations which it should not be able to access after the invalidate and synchronization operations have completed.

Workaround

For updates to the register bits described above:

- Updates to Non-secure register bits listed above must be followed by a **CMD_CFGI_ALL** in the Non-secure command queue, followed by a **CMD_SYNC** in the Non-secure command queue
- Updates to Secure register bits listed above must be followed by a **CMD_CFGI_ALL** in the Secure command queue with SSec = 1, followed by a **CMD_SYNC** in the Secure command queue

For command queue invalidate operations, perform one of the following:

1. Ensure that the device cannot issue transactions that use the invalidated transactions while invalidations are in progress
2. Repeat every **CMD_TLBI_*** and **CMD_CFGI_*** operation before the corresponding **CMD_SYNC** command

For CPU TLBI Inner-Shareable commands that the MMU-700 receives on the DVM interface, perform one of the following:

1. Ensure that the device cannot issue transactions that use the invalidated transactions while invalidations are in progress
2. Repeat every affected TLBI command before the corresponding DSB instruction

2105620
TLB allocation inefficient in certain conditions

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B.
Fault Status: Present in: r0p0, r0p1. Fixed in: r1p0.

Description

There are occupancy counters for managing MPAM and Stage level partitioning in TLBs. These counters are expected to be decremented when invalidations occur. Under certain conditions, this decrement does not occur, and as a result of this allocation in the TLBs does not occur when it is expected to occur.

Conditions

This erratum occurs when MPAM is enabled in any of Main TLB, Walk Cache, and Configuration Cache, or when Stage level partitioning is enabled in Walk Cache. This erratum occurs when any of the following conditions occur:

1. When self invalidation of the entries occur because of a RAS error. This applies to the Main TLB and Walk Cache.
2. When duplicate entries in a set are self-invalidated because of insertion of a new entry that matches several existing entries. This could occur when the translation size of the new entry is larger than existing entries, which happens when the Break Before Make Level 2 feature support is used. This applies to the Main TLB and Walk Cache.
3. When an invalidation that requires walking of the Cache, such as INV_ALL or INV by ASID or INV by VMID is in progress, and this coincides with busy lookup and update traffic to the caches, then the counters in certain sets are not decremented. With INV_ALL, it is possible that up to 1% of occupancy counters in the sets are not decremented in certain worst-case conditions. The impact for INV by ASID or INV by VMID would be much less compared to this.

Implications

When the above conditions occur, the allocation counters become inaccurate, making the allocation of entries into TLB inefficient. The errors accumulate over time, and the TLBs become unusable because nothing can be allocated into them.

Workaround

1. Disable MPAM and Stage Level partitioning. This ensures allocation of entries into TLBs without considering partitioning, and the TLB is usable, similar to previous SMMU products.
    - MPAM in TBU Main TLB can be disabled by setting bit 8 in the TBU_CTRL register.

    - MPAM and Stage level partitioning can be disabled in the Config and Walk cache by setting bits 22, 24, and 26 in the TCU_CTRL register.
2. When TBU or TCU is powered down and powered up again, the counters are reset to 0. Performing this operation at regular intervals can ensure that the errors are not built up, and that the TLB performance remains acceptable.
3. If the PARTID that is associated with streams of transactions is modified on invalidation of that stream of transactions. This ensures that the new PARTID occupancy remains accurate for a longer period of time.

2133013
Stage 1 DVM invalidation operations of translations using TTB1 might not succeed

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

Armv8 Architecture allows for a maximum *Virtual Address Size* (VAS) of either 49 or 53 bits.

MMU-700 implements a VAS of 53 bits, and expects the input address for translation to have full 64-bit valid VA, and the VA presented for TLB invalidation operations received on the ACE-Lite DVM Interface to contain at least 53 valid bits of VA, sign-extended from 49 bits if the processor uses 49 bits of VAS.

Some processor and System IP components implement a VAS of 49 bits and assume global agreement on significant bits of the VA, that is, a global VAS configuration, so do not sign-extend or preserve bits beyond VA[48:0]. In such a case, although MMU-700 treats VA[52:0] as significant, the DVM TLBI invalidation might incorrectly contain VA[52:49] bits as 0.

In this case, the way that the SMMU interprets the address differs from the intention of the sending agent, and the intended address is not invalidated.

Conditions

This issue occurs when all the following conditions occur:

1. The system uses ACE-Lite DVM for invalidation operations that are targeting the TTB1's address space for invalidating TLBs inside the MMU-700
2. The MMU-700 has received an address that requires a Stage 1 Translation, and bit 52 of the Virtual Address of the transaction is set to 1, requiring TTB1 to be used for translation. This translation is cached in the MMU-700 TLBs.
3. The MMU-700 receives a TLBI VA Invalidation operation, aimed at invalidating a range of Virtual addresses, from the ACE-Lite DVM interface, aimed at invalidating the cached stage 1 translation using TTB1 as the base address and the bits [52:49] for the VA that are received are set to 0.

When all the above conditions occur, the cached stage 1 TLB entry is not invalidated as expected.

Note: This description is correct if VA[52:49] is driven to 0, which is the typical case, in all received TLBI invalidations. If those bits are driven to a value other than the sign extension or 0, TTB0 addresses might also be affected.

Configurations affected

This issue occurs in systems with the following configuration:

1. The system contains processors that have a VA size of 49, that is, they do not support the Armv8.2 LPA feature
2. Processors and the interconnect do not perform sign-extension of bits [52:49], when presenting a VA for TLB invalidation on the ACE-Lite DVM Interface.

Note: This affects all current Arm cores and all current Arm interconnect IP.

Implications

Because the intended invalidation operation does not succeed, this results in the incorrect address being accessed post invalidation, and data corruption can occur. This issue does not cause any security violation because Non-secure transactions cannot access Secure locations.

Note: We do not expect systems that are running Linux to be affected because all versions of the upstream SMMU driver, up to at least v5.13, do not perform DVM invalidations to the TTB1 region.

Workarounds

System workaround

In systems that this issue affects, ensure that VA[52:49] presented to the MMU-700 on the ACE-Lite DVM interface are sign-extended from VA[48] when sending TLBI invalidations that specify a VA.

Software workarounds

If the conditions for Erratum 2214515 do not apply, and the conditions for Erratum 2216408 also do not apply, you can use one of the following software workarounds:

1. For invalidations that are intending to invalidate VAs using TTB1, use TLBI_***ASID or TLBI****VMID commands instead of TLBI****VA or TLBI***_VAA commands
2. Use Command Queue for invalidations when invaliding TTB1 addresses instead of the ACE-Lite DVM Interface. If this also requires software to use Command Queue invalidations for TTB0, then we recommend setting SMMU_CR2.PTM and SMMU_SCR2.PTM to 1 to prevent DVM invalidations from being acted on.

If the conditions for Erratum 2214515 apply, avoid using TLBI_***ASID commands and use TLBI****VAA commands instead of TLBI**_VA commands through the command queue.

If the conditions for Erratum 2216408 apply, ensure that any invalidation that uses an address (TLBI_***VA, TLBI****VAA, or TLBI**_IPA) specifies an address range instead of a simple address.
Note : Errata 2214515 and 2216408 are fixed in version r1p1, and so the above two statements do not apply for r1p1 and later versions.

2216408
Invalidated transactions might sometimes complete after SYNC completion

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B.
Fault Status: Present in: r0p0, r0p1, r1p0. Fixed in: r1p1.

Description

A transaction whose translation walk is in progress and is subject to invalidation by *virtual address* (VA) or *intermediate physical address* (IPA) might complete after the corresponding SYNC has been acknowledged as complete.

Conditions

This issue occurs when all the following conditions occur:

- A translation request is being translated using a translation regime that has leaf level at L1 or L2
- 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 that returns a result that is not Leaf
- An Invalidation request by VA or IPA (TLBIVA or TLBIIPA) that uses a simple address instead of an address range is received by the SMMU which matches the translation that is in progress up to any level other than L3
- The SMMU receives a SYNC request before the completion of its non-leaf lookup/walk and the SYNC dependency on other translations in progress is resolved before the waiting lookup/walk completes

When the above conditions occur, SYNC is incorrectly marked as complete before the completion of pending translation walks, and the completion of the transaction as required by the Invalidation received.

Configurations affected

All configurations are affected.

Implications

Because transactions using translations that were invalidated do not complete before the synchronization is complete, incorrect addresses might be accessed, and data corruption can occur. Non-secure transactions cannot access Secure locations because of this issue.

Note: TLB Entries are not incorrectly updated because of this issue.

Software workarounds

Depending on the invalidation operations you are using, you can use one or more of the following workarounds to avoid this issue:

- Replace non-range **TLBI_*_VAA** commands with the same command, but using an address range.
- If Erratum 2268618 affects your use of the MMU-700, for information about **TLBI_*_IPA** commands, see the workarounds in Erratum 2268618.
  If you retain **TLBU_*_IPA** commands, you must specify an address range.
- If Erratum 2268618 does not affect your use of the MMU-700, replace non-range **TLBI_*_IPA** commands with the same command, but using an address range.
- If Erratum 2214515 affects your use of the MMU-700, replace non-range **TLBI_*_VA** commands with **TLBI_*_VAA** commands that use an address range instead of a simple address.
- If Erratum 2214515 does not affect your use of the MMU-700, replace **TLBI_*_VA** commands with either of:
  - **TLBI_*_VA** commands that use an address range instead of a simple address.
  - **TLBI_*_ASID** commands.

2239241
MPAM CMAX register write uses incorrect bits from the write data

Status

- Affects: MMU-700 SMMU - System Memory Mgmt.
- Fault Type: Programmer, Category B.
- Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

MMU-700 implements 8 bits of significant data for MPAM_CMAX registers. This means that bits [15:8] of the write data should be used when writing to these registers. MMU-700 incorrectly uses bits [7:0]. The same bits are used for reads of the registers. The value of those bits is the value used by the design.

This issue affects the following MPAM_CMAX registers in the TCU:

- MPAMCFG_CMAX for both the Walk Cache (RIS==0) and the Config Cache (RIS==1)
- TCU_WC_S*L*_CMAX

This issue affects the following MPAM_CMAX registers in the TBU:

- MPAMCFG_CMAX

Conditions

The issue applies only when MPAM partitioning is active.

Configurations affected

All configurations are affected.

Implications

The value of bits [7:0] in the register write is the value that is used. If those bits have the value 0, no cache space is allocated. In all cases, if the written value does not contain the same value in bits [7:0] as it does in bits [15:8], then the behavior is not the same as the behavior that is intended by the programmer.

Software workarounds

You can work around this erratum by writing the significant bits in bits [7:0] of the write data in addition to, or instead of having them present in bits [15:8].

Arm recommends that for MMU-700 designs that this erratum affects, you set bits [7:0] and [15:8] to be identical so that when this issue is resolved in a future version of the product, the correct behavior is retained.

2268618
TLBI_(R)IPAS2* and CMD_TLBI(_S)_S2_IPA invalidation of nested translations currently in progress might not succeed

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B.
Fault Status: Present in: r0p0, r0p1, r1p0. Fixed in: r1p1.

Description

After the MMU-700 executes an invalidation that specifies an *intermediate physical address* (IPA), some invalidated stage 2 translations might still access translation structures and be cached.

Conditions

This erratum occurs when all the following conditions occur:

- The MMU-700 is performing a translation for a transaction that requires stage 1 followed by stage 2 translation
- The MMU-700 begins executing an invalidation that matches the IPA of one of the stage 2 translations

When the above conditions are met, the translation might access the old value of translation structures or update the cache with entries that are based on the old value of translation structures.

Configurations affected

All configurations are affected.

Implications

Because the intended invalidation operation does not succeed, the invalidated translations might be accessed post-invalidation, and data corruption can occur. After invalidation has completed, an entry might exist in the cache that should not.

Note: Translation responses that are sent to TBUs are correct, including their indication of cacheability, and therefore this issue does not affect translation caching in TBUs.

Software workarounds

You can work around this issue by using one of the following methods, and we recommend that you choose the method that is closest to the way you are currently performing the invalidation:

1. Replace **CMD_TLBI(_S)_S2_IPA** invalidation and accompanied stage 1 invalidations with one **CMD_TLBI(_S)_S12_VMALL** invalidation.
Replace **TLBI_(R)IPAS2*** invalidation and accompanied stage 1 invalidations with one **TLBI_VMALLS12*** invalidation

2. Replace the invalidation command with the following sequence:

- For command queue, the invalidation sequence is:
  - **CMD_TLBI(_S)_S2_IPA**
  - The accompanying stage 1 invalidate **CMD_TLBI_NH_VA** for the full range of VAs translated by this IPA or **CMD_TLBI_NH_ALL**
  - **CMD_SYNC**
- For DVM, the invalidation sequence is:
  - **TLBI_(R)IPAS2***
  - The accompanying stage 1 invalidate **TLBI_RVA(L)E1** for the full range of VAs translated by this IPA or **TLBI_VMALLE1**
  - **DSB**
- If more than one level of stage 2 translation is cached, you must repeat the entire sequence a number of times that is equal to the number of stage 2 translation levels that are cached. If Erratum 2216408 affects your use of MMU-700, also replace the command with one that specifies an address range if it does not already do so.

Note: Disabling some levels of stage 2 caching can reduce the number of times that is required to repeat the invalidation sequence. You can achieve this by writing zero to the **TCU_WC_S2L*_CMAX** registers. For PCIe Gen5 use cases, Arm observed that enabling only L1 for stage 2 provides the best performance, which requires issuing the invalidation sequence once. If two levels of caching are required to meet your requirements, Arm recommends enabling L1 and L2 and issuing the invalidation sequence twice.

2428613
TBU or ATS connection process can sometimes cause deadlock

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Cat B
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

The TBU or ATS connection process can sometimes cause deadlock.

Conditions

This issue can occur in either of the following scenarios, for TBU or ATS. Precise timing is required to trigger this issue because two sets of activity must interact in a particular part of the MMU-700.

TBU

A TBU connection message arrives at the TCU at the same time as a register read or write that is targeting the TBU that is attempting to connect. TBU disconnections do not trigger this issue.

ATS

An ATS connection message arrives at the TCU at the same time as an ATS invalidation message, an ATS synchronization message, or an ATS page response message because of a command that is read from the Non-secure command queue. ATS disconnections do not trigger this issue.

Configurations affected

All configurations.

Implications

TBU

The register access never completes on the APB interface. The APB interface becomes deadlocked, the agent accessing it might also be affected, and extra APB accesses are not possible.

ATS

When a **CMD_PRI_RESP** command or a **CMD_SYNC** command being read from the Non-secure command queue triggers this issue, the command is not acknowledged. As a result, the command is not considered to be consumed and no extra commands are executed from either the Secure or Non-secure command queues.

When a **CMD_ATC_INV** command being read from the Non-secure command queue triggers this issue, that command completes in the command queue, but the next such **CMD_ATC_INV** or **CMD_SYNC** from the Non-secure command queue is not able to progress and therefore no extra commands are executed from either the Secure or Non-secure command queues.

If a **CMD_SYNC** from the Non-secure queue becomes stuck in any way, directly or indirectly, synchronizations from other origins, **CMD_SYNC** from the Secure command queue, DVM synchronizations and internally-generated synchronizations that are associated with updates to some architectural registers, are also unable to progress.

Workaround

TBU

For the case of TBU connections and APB register accesses, you can avoid this issue by ensuring that register accesses are not made to TBUs that are not known to be in the LPI Q_RUN state (and therefore connected) and that remain in that state until the register accesses are complete.

This should align with normal power-management behavior because accesses to TBUs that are not connected would not provide useful read data or have persistence of any writes.

In a system with dynamic power-management, you can determine the connection status of a TBU node by reading the appropriate TCU_NODE_STATUS register for a TBU that is known to be powered on. This enables the software to avoid any race condition that is associated with connections that were caused by the power management system turning a TBU on.

ATS

For the case of ATS connections and Non-secure command-queue commands, you can avoid this issue by ensuring that an ATS connection does not occur while the Non-secure command queue is enabled.

We expect that an ATS connection is often aligned with SMMU power management. The TCU always enters the run state with the Non-secure command queue disabled. When all nodes are known to be connected, the Non-secure command queue can be enabled.
Some third party IP might not initiate a ATS connection on power up. In that case, a specific initiation of the connection might be required to ensure that connection is made before the Non-secure command queue is enabled.

2650791
MPAM monitor values incorrect if configuration changed while enabled

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Cat B.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

The expected use case of the MPAM programming interface is that the control and filter properties are changed only when the monitor is disabled.

The MPAM architecture also supports a use case where the control and filter properties might be updated regardless of the enable status of the monitor, but this support is currently intended only for backwards compatibility.

In affected versions, the backward compatibility support is absent.

If the filter properties of an MPAM monitor are changed while it is enabled, the change does not cause the filter value to be recalculated, and the monitor indicates an incorrect value as a result.

If the control properties of the monitor are changed while the value indicates not ready, the value becomes incorrect.

If the monitor is disabled while the value indicates not ready, the value might become incorrect when the monitor is re-enabled.

Conditions

This can begin to occur at any time if the following are true:

- The *_CTL.EN field of the monitor is HIGH and there is a write to the corresponding *_FLT that changes its value, or
- The *.NRDY field of the monitor is HIGH and there is a write to *_CTL

When the above conditions have occurred, the problem is present continuously.

Configurations affected

All configurations are affected.

Implications

The value of the monitor might be incorrect even if the associated *.NRDY field is LOW.

Workaround

Follow the intended usage mechanism:

- Poll NRDY and wait for it to be 0 before disabling the monitor
- Update the filter and control properties when EN is LOW

2812531
Walk cache updates sent incorrectly after invalidation

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B.
Fault Status: Present in: r0p0, r0p1, r1p0, Fixed in: r1p1.

Description

The SMMU receives a transaction that requires translation table walks to be performed by the *Translation Control Unit* (TCU). If the translation table walk that is in progress encounters two different types of invalidation requests without an intervening invalidation synchronization operation, as the sequences below describe, then the TCU walk cache is incorrectly updated with stale data.

Conditions

This erratum occurs only when a translation that is in progress is affected by an invalidation as the events below specify. If any invalidation is received after corresponding translation walks have completed, then the issue does not occur.

Any of the following sequences of events can result in the erratum:

Event sequence 1:

1. TCU is configured to have Stage 2 Walk Cache entries, and the stage 2 walk cache update is not disabled.
2. TCU receives a translation request that requires Stage 1 followed by Stage 2 Translation and is in the process of performing the translation table walk.
3. TCU receives an invalidation request that affects stage 1 translation only and executes the request.
4. TCU receives another invalidation request that affects Stage 2 translation, without an invalidation synchronization operation between the invalidation operations
5. The Stage 2 invalidation coincides with the TCU performing the Stage 2 translation and is executed before the Stage 2 translation walk cache update is complete.
6. The TCU completes the Stage 2 translation and updates its Stage 2 walk cache with invalidated data potentially violating the Stage 2 invalidation operation.

Or

Event sequence 2:

1. TCU has received a request to perform translation table walk.
2. TCU receives a configuration invalidation request and executes it before the translation request has started accessing translation table data (distinct from configuration data).
3. TCU starts performing the translation table walk.

4. TCU receives another invalidation request without an invalidation synchronization operation between the invalidation operations. The invalidation request affects the stage and level of the walk that the TCU initiates, and is executed before the translation walk is complete.
5. TCU completes the translation in progress and updates the walk cache with the invalidated data, potentially violating the second invalidation operation.

Or

Event sequence 3:

1. TCU is configured to cache non-leaf levels in the Walk Cache.
2. TCU receives a translation request and is in the process of performing the translation table walk.
3. TCU receives a leaf level invalidation for Stage 2 while performing a Stage 2 only translation request, or it receives a leaf level invalidation for a Stage 1 translation.
4. After the first invalidation execution is complete, the TCU receives another invalidation request that affects non-leaf translation levels at the same translation stage as the first invalidation and without an invalidation synchronization operation between the invalidation operations. The invalidation must affect the translation table walk that is in progress.
5. The TCU completes the translation table walk and updates the non-leaf level walk cache entry with invalidated data, potentially violating the second invalidation operation.

Configurations affected

All configurations are affected.

Implications

It is possible that stale translation data is obtained from the walk cache, resulting in incorrect translation, which can result in data corruption.
This erratum does not cause Secure state data to be accessed or overwritten by Non-secure state data.

Workarounds

You can work around this issue by ensuring that an invalidation synchronization operation exists between any of the following pairs of invalidation operations, whether they are adjacent or separated by other commands:

1. A Stage 1 Invalidation that is followed by a Stage 2 invalidation
2. A Configuration Invalidation that is followed by any TLB Invalidation
3. A Leaf invalidation that is followed by a Non-leaf invalidation

Invalidation operations arriving on the DVM interface are interleaved with those being executed from the command queue and neither the SMMU driver software nor the SMMU hardware can control how the DVM and command queue invalidations might be ordered or interleaved. This means that to ensure the workaround achieves the goal stated above, it is also necessary to disable invalidation by DVM, and invalidations must occur only through the command queue.

Category B (rare)

2214515
Global entries are not always invalidated as expected

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B (rare).
Fault Status: Present in: r0p0, r0p1, r1p0. Fixed in: r1p1.

Description

A transaction whose translation walk is in progress, which is marked as a global translation and is subject to invalidation by the following operations is not invalidated as expected, unless the ASID that is specified also matches:

- TLBI_S_EL1_VA
- TLBI_NS_EL1_VA
- TLBI_S_EL2_VA
- TLBI_NS_EL2_VA

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 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 the following operations:
  - TLBI_S_EL1_VA
  - TLBI_NS_EL1_VA
  - TLBI_S_EL2_VA
  - TLBI_NS_EL2_VA
- 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: We do 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. If Erratum 2216408 affects your usage of MMU-700, ensure that any invalidation that the replacement command specifies uses an address range instead of a simple address.

2808817
A device that can issue FIXED bursts can bypass hypervisor Force Write-Back controls to read old memory contents

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category B (rare).
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

When a device issues a transaction with AxBURST = FIXED and the translated memory type and shareability are Normal Write-back Cacheable Shareable, the transaction is converted to Normal Write-back Cacheable Non-shareable, to avoid violating the ACE-Lite protocol. This can result in a device being able to access memory in a non-coherent manner even when a hypervisor is using the S2FWB feature to force all transactions to be Normal Write-back Cacheable Shareable. This can enable the device to access stale data which it should not be able to access.

Conditions

The following steps must occur:

1. The Hypervisor allocates a page of memory to an Operating System, or uses it itself (Owner 1)
2. Owner 1 writes data to the page
3. The caches write the data back to memory
4. The Hypervisor removes access to the page for Owner 1
5. The Hypervisor has the page mapped as cacheable
6. The Hypervisor clears the page by writing zeroes to it, and does not issue cache maintenance operations to force the caches to write the data back to memory, based on the behavior in step 7. The data remains in a local coherent cache, for example, the processor cache.
7. The Hypervisor allocates the page to a new Operating System (Owner 2), and maps it to a device, with the following configuration:
   - STE.S2FWB=1
   - Stage 2 MemAttr[1:0] is `0b10`, forcing Normal Write-Back
   - Stage 2 SH[1:0] is `0b10` or `0b11`, selecting Outer Shareable or Inner Shareable
8. The device issues a read to the page with a transaction with ARBURST=FIXED.

The transaction in step 8 is output with ARCACHE=WriteBack and ARDOMAIN=Non-shareable. In some systems, this enables the transaction to access the data written by Owner 1 in step 2.

Configurations affected

Systems must have all the following properties to be affected:

- The system must include devices capable of issuing FIXED bursts.
  - Devices that issue FIXED bursts are rare and they are not normally generated by PCIe Root Ports.
- The interconnect must support coherency for devices.
- The interconnect must be able to bypass coherency when a transaction has AxDOMAIN=Non-shareable.
  - Many interconnects treat all Cacheable transactions as Shareable regardless of the AxDOMAIN. This includes Arm CMN interconnects up to and including CMN-700 and **~~CI-700~~**.

Implications

In affected systems, a device allocated to one operating system might be able to access data from another operating system when pages are reallocated.

Systems which are impacted by this are likely to be rare.

This erratum affects only usage models where the hypervisor uses S2FWB to force all accesses to a page to be cache-coherent:

- If an operating system sets the stage 1 memory type and shareability to Normal Inner Write-Back Cacheable Outer Write-Back Cacheable Inner or Outer Shareable and a device issues a FIXED burst then this also results in a Non-shareable transaction. This is by design and the operating system must avoid configuring devices to use FIXED bursts when coherency is required.
- If a hypervisor does not use the S2FWB feature, then it cannot force all accesses to be cache-coherent, because if stage 1 translation results in a Non-cacheable memory type, the final memory type is always Non-cacheable or Device.

Software workarounds

When a device can issue FIXED bursts, the hypervisor should not expect S2FWB to prevent that device from bypassing cache coherency. For example, when the hypervisor allocates a page to a device which can issue FIXED bursts, after clearing the page, it must issue cache maintenance operations to clean the caches.

Category C

1768143
Non-zero value of SSID generated in PRI response when SSV is 0

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

DTI mandates that the SSID value that is generated in the DTI PRI response is not 0 when SSV is 0. This is violated under certain conditions.

Conditions

When SSV is 0, SSID is expected to be 0 on the PRI response that is generated from the TCU. If Software provides a non-zero SSID, this is propagated on the DTI.

Configurations affected

All configurations.

Implications

No functional issues are expected from SSID not being 0. This issue is only a DTI protocol violation.

Workaround

No workaround.

1781603
Event queue not overflowing when filled and fault not reported

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

If the event queue in the MMU-700 is set up to be full, and an event is generated, the event queue overflow flag is not set.

Conditions

This issue occurs when all the following conditions occur:

- The event counter is set up to be full, by making producer and consumer pointers the same, and the wrap bit is set
- The event queue is enabled
- A translation results in a non-stallable fault and reports an event to the event queue, and because the event queue is full, the event report is discarded

In these conditions, the event queue overflow flag is expected to be set, but is not set.

Configurations affected

All configurations.

Implications

Because the overflow flag is not set, software is not made aware of the non-stall events being discarded. There is no other functional impact.

Workaround

Ensure that the event queue is not programmed to be full, before it is enabled. This results in an event queue overflow flag being reported correctly.

1784630
Incorrect IPA in fault syndrome reported for HTTU enabled nested translations

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

When the MMU-700 reports a Stage 2 permission fault for denying write permission while performing *Hardware Translation Table Update* (HTTU) in stage 1, incorrect IPA is reported in the fault syndrome.

Conditions

This issue occurs when all the following conditions occur:

- MMU-700 is performing page table walk for a write transaction that requires stage 1 followed by stage 2 translation
- HTTU is enabled for the stage 1 translation by setting of cd.ha or cd.hd bits, and dirty bit marker is set in the stage 1 page table, and stage 1 access permissions indicate Writable clean
- Stage 2 page tables deny write permission, and Stage 2 HTTU of the dirty flag is not enabled
- An earlier read transaction to the same address has resulted in the stage 1 translation being cached in the walk cache
- Stage 1 HTTU update is required for the access flag or the dirty flag, and this results in a stage 2 permission fault being raised
- The IPA reported as part of this fault syndrome is incorrect

Configurations affected

This issue affects configurations where HTTU is enabled through setting the **sup_httu** tie-off signal HIGH.

Implications

In most systems, this issue occurs only if the translation tables are incorrect. This issue does not impact normal translation process and affects only the reported field in the fault syndrome.

Workaround

Because there is no functional issue, in most systems, no workaround is required.

1871067
C_BAD_STE not asserted under certain conditions

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

The MMU-700 does not generate C_BAD_STE when *Hardware Translation Table Update* (HTTU) is enabled when it is not supported, or when VMSPTR in the Context descriptor is expected to be used and out of address range.

Conditions

This issue occurs when any of the following conditions occur:

1. The **sup_httu** tie-off that is connected to the TCU is tied-off to 0, indicating that the system does not support atomic transactions and the Stream table Entry for any stream contains S2HA or S2HD set to 1
2. The Stream table entry for any transaction indicates that stage 1 followed by stage 2 translation is enabled, the S1MPAM field in the stream table entry is 1, indicating that VMSPTR must be used to fetch the PARTID, and if the VMSPTR is out of address range indicated by the **sup_oas** tie-off

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

Configurations affected

All configurations.

Implications

1. For condition 1, software receives a permission fault instead of C_BAD_STE. The HTTU is not expected to be enabled when the system does not support atomic transactions, and therefore this does not have any functional impact in the expected use cases. The TCU ensures that it does not generate atomic transactions in this case.
2. For condition 2, the address that is generated for the VMSPTR fetch has the bits above the supported address range set to 0. This is also not expected to cause any functional issues because the address that is programmed is normally not expected to be above the supported range.

Workaround

No workaround is required.

1877245
RAS error can result in RAS registers read data being corrupted

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

When a single bit RAS error occurs in the MMU-700, it can result in the TBU RAS register read having corrupted data.

Conditions

This issue occurs when the following conditions occur:

- A RAS register read occurs from the TCU which is non-zero. This might be because of a RAS error or the read of the TCU_ERRFR register.
- A RAS register is read from the TBU that is non-zero
- If the initial read is to the TCU_ERRFR register, then the second read must be from any RAS register other than the TBU_ERRFR register
- If the initial TCU register read is not the TCU_ERRFR register, then the second read can be from any non-zero TBU RAS register

When all the above conditions occur, the TBU RAS register read contains corrupted data.

Configurations affected

All configurations.

Implications

This issue does not result in any functional issues. Because of incorrect read data in the TBU RAS register read, this affects the ability of software to profile or debug the RAS errors.

Workaround

After a non-zero RAS register is read from the TCU, perform a read from the TCU_ERRGEN register, while ensuring that the TCU_ERRGEN register data is 0. This ensures that any subsequent reads to the TBU RAS registers are reported correctly.

1879539
Incorrect fault data sometimes reported

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

Under certain conditions, incorrect event data is written to the event queue or an incorrect fault type is returned on the DTI interface.

Conditions

- When SMMU_S_CR0.SIF is changed with transactions in progress, some of the transactions resulting in fault have s2 and nsipa fields reported incorrectly
- When CD.AA64 = 0 and OAS < 40, the fetch address that is reported has bits above 40 set incorrectly

Configurations affected

All configurations.

Implications

There is no functional impact. The impact is to the capability of software to perform debug in these conditions.

Workaround

No workarounds are required.

1912550
Incorrect shareability for Cache Maintenance Operations (CMOs)

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0. Fixed in: r0p1.

Description

Incorrect shareability is generated for *Cache Maintenance Operations* (CMOs) under certain conditions.

Conditions

The issue occurs when all the following conditions occur:

1. The MTCFG field in the stream table descriptor, that the StreamID of the transaction points to is set to 1
2. The Shareability field in the stream table descriptor and Stage 2 page table entry are set to non-shareable
3. One of the following combination of attributes is set:

- The Memattr field in Stream table descriptor and stage 2 page table entry has Non-cacheable set in inner or outer cacheability domains, but not both
- The S2FWB field is set to 1 in STE, Stage 1 followed by Stage 2 translation is enabled, S1 memory attributes set to inner and outer write through, and stage 2 attributes are set to inner and outer non-cacheable

Configurations

All configurations.

Implications

It is not expected that the Stream table and page table shareability is programmed to non-shareable, when the expected shareability of the translated transaction is outer shareable.

When the conditions listed above occur, incorrect shareability on a CMO means that the CMO is not transmitted to the outer shareability domain, and the required effect of the CMO is not achieved in that domain.

Workaround

Ensure that the shareability in *Stream Table Entry* (STE) or *Page Table Entry* (PTE) is set to the expected domain that the CMO is required to reach to avoid the issue.

2093540
TCU Performance Monitoring Event indicating Config Cache Read inaccurate

Status

Affects: MMU-700 SMMU -System Memory Mgmt.
Fault Type: Programmer, Category C.
Fault Status: Present in: r0p0, r0p1. Fixed in: r1p0.

Description

TCU Performance Monitoring Event indicating Config Cache Read is inaccurate.

Conditions

The `0x93` TCU event is counted when the configuration cache read is caused by lookup and update events, and when the Stream ID of the Lookup or update matches the PMU filter that is programmed. This counter is inaccurate for the following reasons:

1. It is counted for all Stream IDs, not only the one used in the PMU filter
2. It includes Invalidation events and MPAM-related reads in the count, apart from the lookup and update events

Implications

This counter event is unusable.

Workaround

When the STE.CONT field is zero, the sum of the TCU events Config cache lookup (`0x92`) and Config Cache miss (`0x94`) equals the `0x93` TCU event. If STE.CONT is non-zero, the sum of the above two events provides an approximate value of the Config cache read event.

2120193
C_BAD_STREAMID not asserted in certain conditions

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category C.
Fault Status: Present in: r0p0, r0p1. Fixed in: r1p0.

Description

The MMU-700 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 of 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, determined by STE.CONT

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 will program the STE.CONT value that exceeds the span of the stream table. It is also not expected that the stream ID that is generated by the device that is connected to the TBU generates an incorrect Stream ID. When 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

No workaround is required.

2288256
Configuration and translation structures might be accessed by prefetch transactions while SMMU(_S)_CR0.SMMUEN=0 and SMMU(_S)_CR0ACK.SMMUEN=0

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category C.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

After the MMU-700 is disabled for a security level with **`SMMU(_S)_CR0.SMMUEN`**=0 and **`SMMU(_S)_CR0ACK.SMMUEN`**=0, the configuration and translation structures might still be accessed by the MMU-700 internally generated prefetch transactions that is caused by STE[121] being set.

Conditions

This erratum occurs when all the following conditions occur:

- The MMU-700 is programmed to enable prefetch with STE[121]=1
- The MMU-700 performs translations using this STE
- Prefetch transactions are prevented from beginning their translation process by full occupancy of some part of the TCU
- The MMU-700 is programmed to disable SMMU for the security level of this STE, invalidation is performed before modifying translation structures in memory and then the MMU-700 is re-enabled for the security level.

When all the above conditions are met, the configuration and translation structures might still be accessed by the prefetch transactions after completion of an update of SMMUEN from 1 to 0, and cache entries using old values of the translation structure might be present after completion of the update of SMMUEN from 0 to 1.

Configurations affected

All configurations that have the parameter **`TCUCFG_PREFETCH_SUPPORTED`**=1 are affected.

Implications

Because software might modify translation structures while SMMUEN is 0, accesses to those structures during that time can result in incorrect entries being stored in the caches of the MMU-700 TCU.

We do not expect that this issue can be observed in practice because Arm recommends that invalidation occurs after translation structure updates in memory instead of before them in this case, as in other cases of translation structure changes.

Software workarounds

You can work around this issue by using one of the following methods:

1. Disable prefetch by setting STE[121:120]=`0b00`.
2. Issue invalidate all for the security level with SMMUEN=0 after software has finished all modifications, and before updating the SMMUEN from 0 to 1. This does not prevent the architectural error that arises because memory access and cache update occur while SMMUEN=0. However, it does prevent any associated functional error.

- If you have disabled the Secure state, issue these invalidation commands in the following order:
  - **`CMD_CFGI_ALL`** with SSec=1
  - **`CMD_TLBI_SNH_ALL`**
  - **`CMD_TLBI_S_EL2_ALL`**
  - **`CMD_TLBI_EL3_ALL`**

- If you have disabled the Non-secure state, issue these invalidation commands in the following order:
  - **`CMD_CFGI_ALL`** with SSec=0
  - **`CMD_TLBI_NSNH_ALL`**
  - **`CMD_TLBI_EL2_ALL`**

2621157
Multiple Events reported to Event queue for a single transaction

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Cat C.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

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 to 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 contents 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.

2626480
PMU event 0x2 (TLB Miss) triggers incorrectly in TCU

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Cat C.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

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. In the TCU this includes any Configuration Cache or Walk Cache lookups. Affected versions only issue the PMU event when there is a complete miss in the Walk Cache on initial lookup. The event can therefore trigger 0, 1 or 2 times for each time a transaction passes through the TCU and so 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.

2674701
Read of TCU_MSMON_CAPT_EVNT register returns data instead of zero

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Cat C.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

The TCU_MSMON_CAPT_EVNT register is used to capture an MPAM monitor value. The register, as a whole, is defined as readable and writable, but all its bits that are currently defined are either RESERVED and should be RAZ/WI or are write-only and should be RAZ. Instead, the bits that are write-only return the last value written.

Conditions

This issue occurs whenever the MSMON_CAPT_EVNT registers in the TCU are read.

Configurations affected

All configurations are affected.

Implications

The data that is returned is the last data that was written to the register and reveals nothing about the:

- State of the TCU
- State of the wider system
- Contents of any memory

Therefore, there is no implication other than non-compliance with the specification.

Workaround

No workaround is required.

2803259
Incorrect merging of S1 Allocation hints

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category C.
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

The *Stream Table Entry* (STE) can be configured to override the incoming memory type to Non-cacheable or a Device type. In this configuration, the SMMU architecture requires that the allocation hints are replaced by Read-Allocate, Write-Allocate, Non-transient. Non-cacheable and Device transactions do not have allocation hints. However, this architecture requirement is necessary because the memory type might change again later in the translation process, which might mean the allocation hints are required.

Because of this issue, if the final translated memory type is a Cacheable type, the allocation hints might not match the hints that the architecture expects.

Conditions

This issue occurs when all the following conditions apply:

- pcie_mode = 0
- STE.MTCFG = 1
- STE.MemAttr is Non-cacheable or a Device type
- STE.ALLOCCFG is either:
  - Configured to Use incoming and an incoming write transaction is Write No-allocate
  - Configured to Use incoming and an incoming read transaction is Read No-allocate
  - Configured to override the allocation hints to anything other than Read-Allocate, Write-Allocate, Non-transient
- Stage 1 or stage 2 translation replaces the memory type with a Cacheable type

Configurations affected

All configurations.

Implications

There are no practical implications to known usage models.

Workaround

When STE.MTCFG = 1 and STE.MemAttr is Non-cacheable or a Device type, set STE.ALLOCCFG = `0b1110`. This setting replaces the incoming allocation hints with Read-Allocate, Write-Allocate, Non-transient, as required by the SMMU architecture.

2900780
Stream table base address not properly aligned when log2size is configured greater than stream ID size

Status

Affects: MMU-700 SMMU - System Memory Mgmt.
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1. Fixed in: r1p2.

Description

When SMMU_(S)*STRTAB_BASE_CFG.LOG2SIZE(STR_LOG2SIZE) is programmed to be greater than 32, the programmed value must be used to align the lower bits of the SMMU*(S)_STRTAB_BASE.ADDR(STR_BASE_ADDR), while the effective value of the STR_LOG2SIZE, which in this case is 32, must be used for the purposes of input StreamID range checking and Stream table index address calculation. The effective STR_LOG2SIZE value, which is 32 is instead used for stream table base address alignment as well.

Conditions

Erratum occurs when all the following conditions occur:

1. Stream table is programmed to be Linear, by setting SMMU_(S)_STRTAB_BASE_CFG.FMT to 0b00
2. STR_LOG2SIZE value must be programmed to be greater than 32.
3. STR_BASE_ADDR must be programmed with any of the bits STR_BASE_ADDR[37:0] to be non-zero.

Implications

In affected systems, incorrect stream table entry will be fetched, and this could result in the transaction ending in a fault erroneously, or an incorrect translation being applied to the transaction.
Please note that programming the STR_LOG2SIZE to be higher than SMMU_SIDSIZE, and programming the stream table base address to incorrectly aligned is not an expected use case. as this programming would result in some of the stream table entries unable to be reached.

Software workarounds

This erratum can be worked around by one of the following workarounds:

1. Programming correctly aligned STR_BASE_ADDR when STR_LOG2SIZE is programmed to be greater than 32.
2. Use a two level stream table instead of a linear stream table.

2945011
Address size fault raised when some RES0 page table descriptor bits have non-zero values

Status

Affects: MMU-700 SMMU - System Memory Mgmt
Fault Type: Programmer, Category C
Fault Status: Present in: r0p0, r0p1, r1p0, r1p1, r1p2. Fixed in: Open

Description

In a system that uses the 64KB granule, when the ARMv8.2-LPA feature is used, MMU-700 is expected to treat bits [41:32] of the Level 1 block descriptor as RES0, even if a non-zero value is written. If those bits are non-zero, the MMU-700 raises an address size fault.

Conditions

This erratum occurs only when all the following conditions occur:

- The system uses a 64KB granule with the ARMv8.2-LPA feature enabling 52 bits of Physical Address
- A Level 1 Page table entry is programmed as a block descriptor, resulting in a 4TB (terabyte) page size
- Any of the bits [41:32] of the page table descriptor are programmed as non-zero

Implications

When any of bits [41:32] are non-zero, an address size fault is raised, and the transaction is blocked from progressing.
Note: There is no known use case for these bits to be programmed to be non-zero values. It is expected that software programs these bits as zero, and this erratum never occurs during the normal course of events.

Software workaround

Writing the page table address bits [41:32] as 0 ensures that the transaction progresses correctly.