CoreLink™ System Memory Management Unit (MMU-500)

Software Developer Errata Notice

This document contains all errata known at the date of issue, in releases up to, and including, revision r2p4.

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.

Product Status

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

Web address

http://www.arm.com/.

Feedback on this product

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

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

Feedback on this document

If you have comments on content then send an e-mail to errata@arm.com giving:

- The document title.
- The document number: ARM-EPM-133458.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

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 on page 8 identifies errata that have been fixed in each product revision.

2 February 2018: Changes in document version 2.0				
ID	**Status**	**Area**	**Cat**	**Summary of erratum**
1047329	New	Programmer	CatB	Prefetch causes incorrect translation when both translation stages are enabled

6 July 2017: Changes in document version 1.0				
ID	**Status**	**Area**	**Cat**	**Summary of erratum**
818719	New	Programmer	CatA	Synchronization Complete is not generated for Synchronization request when SYSBARDISABLE is HIGH
842869	New	Programmer	CatA	Synchronization Complete is not generated when a stalled transaction is present
815719	New	Programmer	CatB	Undetected permission fault when both stage 2 only and stage 1 followed by stage 2 translations are used
815919	New	Programmer	CatB	ACREADY is dependent on page table walk read completion, and this violates AMBA spec
817219	New	Programmer	CatB	Incorrect memory attributes generation during certain page table walks
819119	New	Programmer	CatB	Sideband signal to CCI arqosarb is driven to indeterministic value in some conditions
820469	New	Programmer	CatB	Clock gating request that is accepted by TCU when CRVALID output is HIGH
820471	New	Programmer	CatB	Enabling a cache when an invalidation is in progress results in stale cache data
826419	New	Programmer	CatB	Incorrect prefetch in V7S mode
837019	New	Programmer	CatB	TBU not asserting qactive_<tbucfg_cg> on power up request out of reset
841119	New	Programmer	CatB	Updating a translation entry to increase page size might cause data corruption
562869	New	Programmer	CatB	Invalid AXI R-channel byte lanes used on PTW interface for pre-fetching VMSA descriptors
752357	New	Programmer	CatB	MMU-500 does not invalidate Global Stage 1 Page Table Entries in certain conditions

818688	New	Programmer	CatB	SMMU_IDR0.BTM is 1 when BTM not supported by system
837736	New	Programmer	CatB	MMU-500 does not invalidate TTBR1 TLB entries in some conditions
842870	New	Programmer	CatB	Starvation of entries waiting in PTW Queue, when PTW Queue is full
809669	New	Programmer	CatC	Incorrect SMMU_CBn_FSYNR0.NSATTR logged for Secure contexts
809671	New	Programmer	CatC	Allocate hint set incorrectly for some programming and input combinations
814019	New	Programmer	CatC	Final stage 2 walk in a nested translation not cached in IPA2PA cache
820470	New	Programmer	CatC	Context fault register write can be prevented by transactions that continuously fault
820472	New	Programmer	CatC	Bit 48 in Fault Address Register not updated in Stage 2 only configurations
821769	New	Programmer	CatC	Starvation of a TBU leads to system delays
821770	New	Programmer	CatC	sACR.SMMU_PAGESIZE change results in a spurious update to the context bank0 register
825670	New	Programmer	CatC	SMMU_IDR1.NUMS2CB is incorrect in Stage 2 only configurations
829921	New	Programmer	CatC	MAJOR version in IDR7 incorrect

Errata summary table

The errata associated with this product affect product versions as below.

ID	Cat	Summary	Found in versions	Fixed in version
818719	CatA	Synchronization Complete is not generated for Synchronization request when SYSBARDISABLE is HIGH	r0p0	r2p0
842869	CatA	Synchronization Complete is not generated when a stalled transaction is present	r0p0	r2p4
815719	CatB	Undetected permission fault when both stage 2 only and stage 1 followed by stage 2 translations are used	r0p0	r1p0
815919	CatB	ACREADY is dependent on page table walk read completion, and this violates AMBA spec	r0p0	r1p0
817219	CatB	Incorrect memory attributes generation during certain page table walks	r0p0	r1p0
819119	CatB	Sideband signal to CCI arqosarb is driven to indeterministic value in some conditions	r0p0	r2p0
820469	CatB	Clock gating request that is accepted by TCU when CRVALID output is HIGH	r1p0	r2p0
820471	CatB	Enabling a cache when an invalidation is in progress results in stale cache data	r0p0	r2p0
826419	CatB	Incorrect prefetch in V7S mode	r0p0	r2p2
837019	CatB	TBU not asserting qactive_<tbucfg_cg> on power up request out of reset	r0p0	r2p2
841119	CatB	Updating a translation entry to increase page size might cause data corruption	r0p0	r2p2
562869	CatB	Invalid AXI R-channel byte lanes used on PTW interface for pre-fetching VMSA descriptors	r0p0	Not fixed
752357	CatB	MMU-500 does not invalidate Global Stage 1 Page Table Entries in certain conditions	r0p0	Not fixed
818688	CatB	SMMU_IDR0.BTM is 1 when BTM not supported by system	r0p0	Not fixed
837736	CatB	MMU-500 does not invalidate TTBR1 TLB entries in some conditions	r0p0	Not fixed
842870	CatB	Starvation of entries waiting in PTW Queue, when PTW Queue is full	r0p0	r2p4
1047329	CatB	Prefetch causes incorrect translation when both translation stages are enabled	r0p0	Not fixed
809669	CatC	Incorrect SMMU_CBn_FSYNR0.NSATTR logged for Secure contexts	r0p0	r1p0
809671	CatC	Allocate hint set incorrectly for some programming and input combinations	r0p0	r1p0
814019	CatC	Final stage 2 walk in a nested translation not cached in IPA2PA cache	r0p0	r1p0
820470	CatC	Context fault register write can be prevented by transactions that continuously fault	r0p0	r2p0
820472	CatC	Bit 48 in Fault Address Register not updated in Stage 2 only configurations	r0p0	r2p0

821769	CatC	Starvation of a TBU leads to system delays	r0p0	r2p1
821770	CatC	sACR.SMMU_PAGESIZE change results in a spurious update to the context bank0 register	r0p0	r2p0
825670	CatC	SMMU_IDR1.NUMS2CB is incorrect in Stage 2 only configurations	r0p0	r2p1
829921	CatC	MAJOR version in IDR7 incorrect	r1p0	r2p0

Errata descriptions

Category A

818719

Synchronization Complete is not generated for Synchronization request when SYSBARDISABLE is HIGH

Status

Affects: MMU-500 SMMU -System Memory Mgmt - TERM

Fault Type: Programmer, Category A

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0.

Description

The MMU-500 supports TLB maintenance operations through the DVM interface of the ACE-Lite port that the page table walker uses, and through the register programming interface.

It supports Invalidation and Synchronization of the invalidation operations as part of the TLB Maintenance operations.

The MMU-500 initiates a Sync Operation when one or more of the following occur:

- A write to the SMMU_sTLBGSYNC register.
- A write to the SMMU_CBn_TLBSYNC register.
- A DVM Sync command is received.

When the MMU-500 has completed the Sync Operation, it must signal this completion through as appropriate:

- Setting SMMU_sTLBGSTATUS.ACTIVE to 0.
- Setting SMMU_CBn_TLBSYNC.ACTIVE to 0.
- Issuing a DVM Complete message.

The **SYSBARDISABLE** is a pin input to each TBU that decides whether the MMU-500 adds a barrier when receiving a Synchronization request for the respective TBU.

When the **SYSBARDISABLE** pin is LOW, the MMU-500 generates barriers that ensure that all transactions that are affected by the Synchronization have completed.

If the **SYSBARDISABLE** pin is HIGH, the MMU-500 does not issue barriers, but counts outstanding transactions in the downstream system. However, in this scenario, the MMU-500 erroneously might not recognize the completion of the Synchronization operation until the upstream system is quiescent for the respective TBU.

As a result, the Synchronization completion might be delayed for longer than expected, or indefinitely delayed by a continuous stream of transactions from the upstream system.

Configurations affected

All configurations.

Conditions

- **SYSBARDISABLE** is connected to 1, and
- There are one or more invalidation requests when there is at least one transaction being processed that is affected by one or more of the invalidation requests, and
- A SYNC request, generated by writing to SMMU_sTLBGSYNC, SMMU_CBn_TLBSYNC, or by a DVM Sync message, has been issued when there are accesses being processed in the MMU-500, and
- There is a continuous stream of transactions being presented to the MMU-500

Implications

When the programming interface is used to sync the invalidation requests:

- The corresponding TLBSTATUS register might only indicate inactive later than the actual completion of the previous accesses, or might not be updated at all.

When the DVM Message Interface is used to sync the invalidation requests:

- DVM Complete might be sent later than the actual completion of previous accesses or DVM Complete might not be sent at all. This might prevent the core that is executing a DSB from completing the DSB.

Workaround

No workaround.

842869

Synchronization Complete is not generated when a stalled transaction is present

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category A

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2. Fixed in r2p4.

Description

The MMU-500 supports TLB maintenance operations through the DVM interface of the ACE-Lite port that the page table walker uses, and through the register programming interface.

It supports Invalidation and Synchronization of the invalidation operations as part of the TLB Maintenance operations. It also supports Stall fault model for its faulted transactions, where the faulted transactions in the stall mode, or stalled transactions, must be explicitly resumed or terminated by programming the appropriate value to the SMMU_CBn_RESUME.TnR field.

The MMU-500 initiates a Sync Operation when one or more of the following occur:

- A write to the SMMU_sTLBGSYNC register.
- A write to the SMMU_CBn_TLBSYNC register.
- A DVM Sync command is received.

When the MMU-500 has completed the Sync Operation, it must signal this completion through as appropriate:

- Setting SMMU_sTLBGSTATUS.ACTIVE to 0.
- Setting SMMU_CBn_TLBSYNC.ACTIVE to 0.
- Issuing a DVM Complete message.

When a TBU receives a SYNC request after a set of INV requests, it waits for all transactions that are affected by the INV requests to be completed to generate a SYNC ACK. The TBU erroneously waits for any transactions that have faulted in Stall Mode to be completed before providing the SYNC Acknowledge to TCU. The TCU does not complete SYNC on the DVM or programming interfaces until SYNC Acknowledge have been received from all TBUs.

This results in deadlock, because the processing entity waiting for SYNC complete might be the one that must issue the Resume on the TCU Programming interface.

The **SYSBARDISABLE** is a pin input to each TBU that decides whether the MMU-500 is to add a barrier when it receives a Synchronization request for the respective TBU.

Configurations affected

All configurations

Conditions when SYSBARDIABLE is HIGH

- There are one or more transactions that have been faulted in stall mode, and
- A SYNC request, generated by writing to SMMU_sTLBGSYNC, SMMU_CBn_TLBSYNC, or by a DVM Sync message, has been issued when the stalled transaction is not yet resumed

Conditions when SYSBARDIABLE is LOW

- Conditions for **SYSBARDISABLE** being HIGH are present and
- There is a transaction newer than the stalled transaction in the TBU that has been affected by Invalidation, and must be completed before providing SYNC Acknowledge

Implications

When the programming interface is used to sync the invalidation requests:

- The corresponding TLBSTATUS register might only indicate inactive later than actual completion of previous accesses, or might not be updated at all.

When the DVM Message Interface is used to sync the invalidation requests:

- DVM Complete might be sent later than the actual completion of the previous accesses, or DVM Complete might not be sent at all. This might prevent the core that is executing a DSB from completing the DSB.

Category A (rare)

There are no errata in this category.

Category B

815719

Undetected permission fault when both stage 2 only and stage 1 followed by stage 2 translations are used

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0. Fixed in r1p0.

Description

The MMU-500 supports stage1 with stage 2 bypass translations, stage 2 only translations, and stage 1 followed by stage 2 translations, also referred to as a nested translation. A nested translation translates an input Virtual Address (VA) to an Intermediate Physical Address (IPA) and then translates that IPA to output a Physical Address (PA). For nested translations, the MMU-500 has a context for stage 1 translation, that points to a different context for the stage 2 translation. The stage 2 context can be simultaneously used for a stage 2 only translation. In a system where the stage 2 context is used both as part of nested translation and for a stage 2 only translation, the result of a stage 2 context page table walk is cached as part of the macro-TLB or a prefetch buffer, that can be used during the nested translation process.

When the stage 2 translation result is cached in the macro-TLB or prefetch buffer during the nested translation, permission checks are not applied on the result that is retrieved from the macro-TLB or prefetch buffer.

Configurations affected

Translation option chosen as nested, that is, stage 1 translation followed by stage 2 translation.

Conditions

Stage 2 translation result is cached in the macro-TLB or prefetch buffer during the nested translation

Implications

Because the translation of the transaction uses a page table without sufficient permission, the end transaction is completed without sufficient permission. This could corrupt system memory.

Workaround

Set the SMMU_CBn_ACTLR.CPRE and SMMU_CBn_ACTLR.CMTLB bits to 0 to disable the macro-TLB and prefetch buffer for all stage 2 contexts to work around the erratum.

-

815919

ACREADY is dependent on page table walk read completion, and this violates AMBA spec

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0. Fixed in r1p0

Description

The MMU-500 supports TLB maintenance operations through the DVM interface of the ACE-Lite port that the page table walker uses. The MMU-500 has an internal 8-deep queue for storing and processing the TLB maintenance operations that are received. The MMU-500 also has an internal queue of transactions performing page table walks as part of the translation process. If the TLB maintenance operations received affect a transaction performing a page table walk, then the MMU-500 creates a dependency between completion of that page table walk and extra processing of TLB maintenance operations. This results in a build-up of commands in the TLB maintenance queue and the **ACREADY** output from the MMU-500 is pulled LOW when this queue is full.

Systems affected

Systems that use DVM for TLB maintenance operations are affected. Systems using only register-based TLB maintenance operations are not affected.

Conditions

When the ACREADY output from the MMU-500 is pulled LOW, with the events described below, then it results in a deadlock.

- An external interconnect component depends on the **ACREADY** output from the MMU-500 being HIGH to complete the page table walk.

  Note: The CCI-400 has this dependency.
- A page table walk for affected transactions does not complete before the internal TLB maintenance queue becomes full.

Implications

The system is deadlocked.

Workaround

This issue can be worked around by setting the SMMU_sCR0.PTM bit to '1' and managing the TLB maintenance through register programming. The monitor and the hypervisor respectively control the SMMU_SCR0 and SMMU_CR0 registers..

817219

Incorrect memory attributes generation during certain page table walks

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0. Fixed in r1p0

Description

The MMU-500 generates attributes for the translated transactions and page table walks using the information that is programmed in the internal registers and the page tables. In this case, the MMU-500 generates the page table walk attributes using only SMMU_CBn_TTBCR register information, and does not combine the attributes from the SMMU_CBARn register.

Configurations affected

Translation option chosen is nested, that is, stage 1 translation followed by stage 2 translation.

Conditions

The erratum occurs when the MMU-500 is programmed to perform stage 1 translation with stage 2 bypass and if the SMMU_CBARn that controls the stage 2 attribute-only transform has one or both of the following stronger than the one specified in SMMU_CBn_TTBCR:

- Memory type.
- Shareability transform.

Implications

This erratum results in incorrect memory attributes generated on cache, PROT, and domain lines. The security attribute is not affected, but other attributes are affected. Incorrect generation of page table walk attributes can mean that the MMU-500 or the core read stale page table data. Using stale page table walk data can result in loss of permission checks or data corruption of the address space that the page table manages. If the shareability is mismatched, then potentially, data corruption of the page table walk data itself is possible.

Workaround

Program the SMMU_CBn_TTBCR register with the final attributes that software requires.

819119

Sideband signal to CCI arqosarb is driven to indeterministic value in some conditions

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0

Description

The MMU-500 performs page table walks to translate the transactions that are received at its TBU slave interfaces. A head-of-line transaction on the port used by the page table walker can block a page table walk transaction that has a higher priority. To hint to the interconnect, such as the CCI-400r1, that this unaccepted head-of-line transaction should be promoted to a higher priority, the MMU-500 exports a signal, **arqosarb**, that is the instantaneous highest priority of any blocked page table walk transaction.

The MMU-500 can be configured to share the PTW interface with the ACE-Lite master interface of TBU0. When there is no active page table walk read transaction pending from the TCU, the value of **arqosarb** driven out for those transactions can be arbitrary, and might result in priority promotion of transactions from TBU0. This unwanted priority promotion might occur when the value on **arqosarb** is higher than the **ARQOS** value on the TBU0 transaction.

Configurations affected

This erratum occurs only when the PTW Interface is shared with the TBU0 master interface.

Conditions

When there is no active page table walk read transaction pending from the TCU.

Implications

The *Quality of Service* (QoS) of the system could be affected because the priority of several transactions might be arbitrarily increased.

Software workaround:

Bit 4 in the Control Override register in CCI-400r1 can be set to 1 to ignore the value that is driven on **arqosarb**. The CCI Control Override register can only be programmed from the Secure side.

Hardware workaround:

Each bit in the **arqosarb** signal can be ANDED with "arvalid & arid[MSB] & ~arbar[0]" to ensure only transactions corresponding to the TCU have a valid value on **arqosarb**.

820469

Clock gating request that is accepted by TCU when CRVALID output is HIGH

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r1p0. Fixed in r2p0

Description

The MMU-500 consists of a Translation Control Unit (TCU) that supports clock-gating using a Q-Channel interface. When the **qreqn_tcu** input signal to the MMU-500 is driven LOW, indicating a request to gate the clock to the TCU, **qacceptn_tcu** is expected to assert LOW only after all current internal transactions are complete. In some conditions, the MMU-500 asserts **qacceptn_tcu** LOW when the **crvalid_ptw** output from the TCU is still HIGH.

Configurations affected

Configurations where the TCU internal clock is chosen to be clocked at half the external clock speed.

Conditions

This erratum occurs when all the following conditions occur:

- All input interfaces to TCU are quiescent.
- All output interfaces from TCU are quiescent except for the CR Channel.
- All of internal TCU state is Idle, that is, there are no page table walks pending, the Programming interface state machine is IDLE and both TCU-TBU and TBU-TCU interfaces are IDLE.
- The input signal **qreqn_tcu** is LOW, output from the TCU **crvalid_ptw** is HIGH, and **crready_ptw** is LOW for more than two clocks.

Implications

After accepting the clock gating request by asserting **qacceptn_tcu** LOW, the clock to the TCU is turned off. This results in the **crvalid_ptw** output from the TCU being continuously HIGH, violating the AXI protocol. This could result in unusual behavior in the interconnect that is connected to the TCU.

Workaround

If the system can ensure that the clock is not turned off when **crvalid_ptw** from the TCU is HIGH, then the issue can be prevented.

There is no software workaround.

820471

Enabling a cache when an invalidation is in progress results in stale cache data

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0

Description

The MMU-500 has a Translation Control Unit (TCU) that contains several caches and TLBs:

- Macro-TLB.
- Prefetch buffer.
- Page table walk cache.
- IPA-to-PA cache.

These caches are enabled by setting to one in the following global control registers:

- SMMU_sACR.S2WC2EN.
- SMMU_sACR.S1WC2EN.
- SMMU_sACR.IPA2PA_CEN.

OR by setting to one the context bank control registers:

- SMMU_CBn_ACTLR.CPRE.
- SMMU_CBn_ACTLR.CMTLB.

The intent of the context bank control registers is to provide finer-grained control of caching for information from different context banks. For example, context banks that serve traffic that is low bandwidth or can tolerate high latency could have caching turned off.

By default, all caching is turned on and it is expected that this is a reasonable choice for most systems.

Invalidations of these caches can be triggered by receipt of an appropriate DVM message, or by writes to explicit TLB invalidation control registers, or by writes to other control registers that cause an implicit invalidation for implementation reasons.

If the cache state is changed from disabled (0) to enabled (1) while, for any reason, an invalidation is occurring, then the invalidation of that cache is not completed properly leaving stale data in the cache.

Invalidations by the Secure world and Non-secure world are treated independently so there is no mechanism for the Non-secure world to interfere with an ongoing Secure-world invalidation.

Implications

When the erratum occurs, the stale data in the caches can lead to virtualization holes and data integrity issues. It cannot lead to a security violation. However, the Secure side can directly change the Non-secure caching bits or initiate Non-secure invalidations while the Non-secure world is changing the caching bits. However, it is not expected that the Secure world would be directly changing the Non-secure world.

Workaround

Software should not alter the following unless the MMU-500 is quiescent:

- SMMU_sACR.S2WC2EN.
- SMMU_sACR.S1WC2EN.
- SMMU_sACR.IPA2PA_CEN.
- SMMU_CBn_ACTLR.CPRE.
- SMMU_CBn_ACTLR.CMTLB.

By default, these bits are enabled and this is reasonable. The recommended software workaround is to not change these bits in any software. The hypervisor must prevent any guest OS from changing the SMMU_CBn_ACTLR of any context banks that it controls. Therefore, the hypervisor must trap any write accesses to the context banks under control of a guest OS. It should then emulate any write that does not change the SMMU_CBn_ACTLR.

One of the software workarounds for erratum 826419 "Incorrect prefetch in V7S mode", indicates that the prefetch for V7S contexts must be turned off by setting SMMU_CBn_ACTLR.CPRE == `0b0`. Therefore, the hypervisor should either ensure that the prefetch for V7 Contexts always stays off, or use the other workarounds indicated in the erratum 826419.

826419

Incorrect prefetch in V7S mode

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1. Fixed in r2p2.

Description

The MMU-500 performs page table walks to translate transactions that it receives at its TBU slave interfaces.

For the V7 Short Descriptor (V7S) context banks, Large and Small pages access a second level page table. The L2 page table covers a 1MB area of VA space, and the table itself is 1KB in size.

The MMU-500 attempts to prefetch descriptors past the descriptor that it currently requires for the translation so that streaming traffic has lower latency.

This erratum means that the MMU-500 might incorrectly prefetch a single descriptor past the 1KB boundary of the L2 table.

For this erratum to occur, all of the following conditions must occur:

- The MMU uses a Short descriptor table format.
- Prefetch is enabled (SMMU_CBn_ACTLR.CPRE == `0b1` which is the default value out of reset).
- The L2 table is not the last 1 KB in a 4KB-aligned unit of memory.
- The data after the L2 table in memory describes a valid descriptor.

Implications

If the conditions that the erratum describe occur, and the prefetched descriptor is interpreted as valid, then it is cached in the prefetch buffer and can be used to perform a translation. This data might not describe a correct translation and might enable the device to access data that it is not permitted to. This might lead to data corruption and information leakage from one device to another in the same Virtual Machine (VM) if there is a stage 2 translation. If there is no stage 2, then the device might access any memory location that is permitted by that security state.

For a Secure context, if the incorrectly fetched data is interpreted to point to a Non-secure memory location then the Secure side could access an incorrect Non-secure memory location, rather than access an incorrect Secure memory location. This might expose a Secure OS to the Non-secure side and so alter the execution of the Secure OSs.

This erratum does not permit one VM to access the data of another VM.

Workaround

Use any one of the following to work around this erratum.

1. Use a V7 long descriptor instead of a short descriptor. This can be enforced using a hypervisor by performing all of the following steps
    a. :Ensure that the guest OS observes SMMU_IDR0.PTFS as `'b01` indicating that the V7 short descriptor format is not supported by the SMMU.
    b. When the context is in AArch32 mode, the hypervisor must ensure that the guest OS reads the bit field SMMU_CBn_SCTLR.EAE as the constant `0b1` and that when the guest OS writes to the physical SMMU the hypervisor forces it to be `0b1`.
    c. When the context changes from AArch64 to AArch32, the hypervisor writes a 1 to the SMMU_CBn_SCTLR.EAE bit.
2. The OS ensures that after every V7 Short descriptor (V7S) L2 page table, that is not at the end of a 4KB, there is at least one single zero word after the 1KB boundary.
3. The OS or the hypervisor turns off prefetch for V7S contexts by setting SMMU_CBn_ACTLR.CPRE == `0b0`. However, erratum 820471 "Enabling a Cache when an Invalidation is in progress results in Stale Cache data" means that when prefetch has been disabled, it cannot be safely re-enabled until the MMU-500 is quiescent. The software workaround for erratum 820471 suggests that the hypervisor might inhibit a guest OS from altering these bits. The hypervisor must ensure that V7S contexts have the prefetch operation disabled and must only re-enable the prefetch when the context changes to V7L/V8L when the MMU-500 is quiescent.

The MMU-500 r2p0 version contains the extra register bits SMMU_SACR.CACHE_LOCK and SMMU_ACR.CACHE_LOCK that can lock the write access to the SMMU_CBn_ACTLR. The Secure monitor can use SMMU_sACR.CACHE_LOCK to ensure a Secure or Non-secure guest OS does not enable the prefetch in V7 contexts with short descriptor. The hypervisor can use the SMMU_ACR.CACHE_LOCK to ensure that a Non-secure guest OS does not enable the prefetch in V7 contexts with short descriptor.

If the Secure world has not set SMMU_SACR.CACHE_LOCK, then the hypervisor can use the SMMU_ACR.CACHE_LOCK to ensure that a Non-secure guest OS does not enable the prefetch in V7 contexts with short descriptor. If the Secure world has set SMMU_SACR.CACHE_LOCK with prefetch enabled then only 1) and 2) are viable workarounds.

837019

TBU not asserting qactive_<tbucfg_cg> on power up request out of reset

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1. Fixed in r2p2.

Description

The MMU-500 does not drive **qactive_<tbucfg>_cg** HIGH on reset, when **qreqn_<tbucfg>_pd** is set.

Configurations affected

This erratum occurs in all configurations.

Conditions

The MMU-500 contains a configurable number of *Translation Block Units* (TBU). Each TBU is instantiated in the clock and power domain of the peripheral to which it is connected. Each TBU connects to the central Translation Control Unit (TCU) through an asynchronous bridge. To support power isolation, the TBU has two low-power channels, a power-gating channel, and a clock-gating channel.

When reset, the TBU initializes to its powered down state. When the **qreqn_<tbucfg>_pd** signal for that TBU is driven HIGH, the TBU generates a power up request to the TCU. This power up transaction requires the clock to be present for that TBU, so the TBU asserts **qactive_<tbucfg>_cg** HIGH, to enable the clock controller to provide the clock.

The TBU handles this sequence correctly in all conditions, except for when it comes out of reset. When the TBU comes out of reset, the **qactive_<tbucfg>_cg** signal is LOW until it receives a clock edge, or it receives a valid transaction on any of its other interfaces. After the TBU receives the first clock edge, it asserts the value **qactive_<tbucfg>_cg** correctly until power is gated to the TBU.

Implications

If the clock controller that provides the clock for the TBU depends only on the **qactive_<tbucfg>_cg** signal for driving **qreqn_<tbucfg>_cg** HIGH, and providing a clock on reset, then the TBU can never leave the power down state.

Workaround

There is no software workaround.

For a hardware workaround, drive **qreqn_<tbucfg>_cg** HIGH, when the TBU leaves reset.

841119

Updating a translation entry to increase page size might cause data corruption

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1. Fixed in r2p2.

Description

The MMU-500 supports the prefetching of translation table entries when this function is enabled. The MMU-500 performs a prefetch only when the page table points to a non-contiguous page descriptor. This means that the page size is 4K in V7 short descriptor format, or it is a page size of 4K in V8 4K granule, or a page size of 64K in 64K granule.

When the MMU-500 prefetches a translation table entry, and if the page size of the prefeched translation table entry has changed, then the MMU-500 updates the cache with incorrect translation table attributes that do not correspond to the old or the new entry.

Conditions

This erratum occurs when all the following sequence of events occurs:

1. Prefetch is enabled for the context in the MMU-500 that is used for translation.
2. The Page Table for a certain VA that is originally programmed to have a 4K or a 64K Page size. If the page is programmed to have a contiguous hint with 4K page size, then this problem does not occur.
3. The VA does not fall in the final two pages of a 2M block. That is for a 4K page size, bits 20:12 of Address are between 0x000 and 0x1FD, and for a 64K Page size, bits 20:16 are between 0x00 and 0x1D.
4. The MMU-500 fetches the page table for that VA and either the page table for VA+4K address in case of 4K page size, or the page table for VA+64K for 64K page size, and then stores it in the prefetch buffer.
5. Software changes the Page size from the page descriptor to a block descriptor.
6. The MMU-500 receives an access that falls within the next page of the original VA. This results in a prefetch buffer hit and a further automatic prefetch access.
7. When performing the walk for the further prefetch access, there is a Walk Cache Miss.

   The combination of a previous prefetch-buffer hit, and Walk Cache miss in this access can occur in one of the following conditions:

   - Walk Cache is turned OFF, and Prefetch is turned ON. This is an unusual combination, and ARM does not expect this to occur.
   - Walk Cache is on, but Walk Cache entry for the VA has been replaced, and the prefetch buffer has not been replaced. Because the size for Walk Cache and Prefetch buffer is the same, one of the following conditions can evict the Walk Cache, when keeping the Prefetch buffer entry:
     - Four accesses that return a proper page table for normal access, but a faulty entry for Prefetch.
     - Walk Cache entries have been subjected to invalidation, but the Prefetch buffer has not yet been subjected to invalidation, providing software uses invalidation using the 'VA' approach.
8. When this prefetch returns a block descriptor instead of a page descriptor, the result is a prefetch buffer update with incorrect attributes. These attributes are random, and are not necessarily part of either the old page table attributes or new page table attributes.
9. If the MMU-500 receives an access whose VA falls in the second page after the original VA, but before the invalidation for that prefetched location is received, then the MMU generates an access with incorrect memory attributes, or generates an access is that is not normally permitted.

Implications

When this erratum occurs, the MMU-500 generates one or both of an incorrect physical address and incorrect memory attributes.

Data corruption can occur because of the incorrectly generated physical address.

The consequences of the incorrect Memory attribute generation are the following:

- The MMU-500 could generate an illegal transaction. This can result in data corruption.
- A transaction can be generated with incorrect cacheability attributes and shareability attributes. This can create coherency issues if a system cache is present downstream of the MMU-500 and can result in data corruption.

It is possible for a Guest OS to create the conditions that are required for this issue and cause data corruption. A Guest OS can cause data corruption in other Guest OSes or Hypervisor address space. However, a Guest OS cannot cause

data corruption in a Secure address space.

Workaround

This erratum can be worked around by turning Prefetch off.

The OS or the hypervisor can turn off Prefetch for all contexts by setting SMMU_CBn_ACTLR.CPRE == `0b0`. The hypervisor must prevent any guest OS from changing the SMMU_CBn_ACTLR of any context banks that it controls. Therefore, the hypervisor must trap any write accesses to the context banks that are under the control of a guest OS. It then emulates any write that does not change the SMMU_CBn_ACTLR.

The MMU-500 r2p0 version has the additional register bits SMMU_SACR.CACHE_LOCK and SMMU_ACR.CACHE_LOCK that can lock the write access to the SMMU_CBn_ACTLR. The Secure monitor can use SMMU_sACR.CACHE_LOCK to ensure that a Secure or a Non-secure guest OS does not enable the prefetch in any context. The hypervisor can use SMMU_ACR.CACHE_LOCK to ensure that a Non-secure guest OS does not enable the prefetch in any context. If the Secure world has not set SMMU_SACR.CACHE_LOCK, then the hypervisor can use SMMU_ACR.CACHE_LOCK to ensure that a Non-secure guest OS does not enable the prefetch in its context.

The probability of this erratum occurring can be reduced greatly by always keeping the Walk Cache on when Prefetch is turned on.

562869

Invalid AXI R-channel byte lanes used on PTW interface for pre-fetching VMSA descriptors

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2, r2p4.

Description

The MMU-500 performs page table walks to translate transactions that it receives at its TBU slave interfaces.
The MMU-500 attempts to prefetch descriptors past the descriptor that it currently requires for the translation so that streaming traffic has lower latency.

For the V7 Short Descriptor (V7S) context banks, this erratum means that the MMU-500 might store Garbage data in the prefetch buffer

Conditions

For this erratum to occur, all of the following conditions must occur:

- The MMU uses a Short descriptor table format.
- Prefetch is enabled (SMMU_CBn_ACTLR.CPRE == `0b1` which is the default value out of reset).
- The current page table entry being fetched is at an address offset of 0x4.
- A data width of 128 is used for the TCU.

When the above conditions occur MMU-500 initiates an unaligned burst, with address offset of 0x4, with a length of 8 bytes. With this transaction, the system must return valid data only on the 0x4 to 0x7 byte lanes. However, the MMU-500 also samples byte lanes 0x8 to 0xB. There is no issue if the system returns the data from the Memory in these byte lanes. However, if the system returns some random data, which seem to describe a valid descriptor, then the MMU-500 stores that random data in the prefetch buffer. When a transaction that hits the prefetch buffer is received, this results in a data integrity issue.

Implications

If the conditions described in the erratum description occur, and the prefetched descriptor is interpreted as valid then it is cached in the prefetch buffer and could be used to perform a translation. This data might not describe a correct translation and might allow the device to access data that it is not allowed to. This might lead to data corruption and information leakage from one device to another in the same Virtual Machine (VM) if there is a stage 2 translation. If there is no stage 2, then the device might access any memory location that is allowed by that security state.

For a Secure context, if the incorrectly fetched data is interpreted to point to a Non-secure memory location then the Secure side could access an incorrect Non-secure memory location, rather than access an incorrect Secure memory location. This might expose a Secure OS to the Non-secure side and so alter the execution of the Secure OSs.

This erratum does not permit one VM to access the data of another VM.

Workaround

Use any one of the following to work around this erratum:

- Use a V7 long descriptor instead of a short descriptor.
- The OS or the hypervisor turns off prefetch for V7S contexts by setting SMMU_CBn_ACTLR.CPRE == `0b0`.

752357

MMU-500 does not invalidate Global Stage 1 Page Table Entries in certain conditions

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2, r2p4.

Description

The MMU-500 supports TLB maintenance operations through the DVM interface of the ACE-Lite port that the page table walker uses, and through the register programming interface.

When the MMU-500 receives a Stage 1 invalidation request by Virtual Address that requires the ASID value to be matched, and if any of the TLB entries that must be invalidated are marked as 'Global', that is, nG bit in the Page Table Leaf or Block entry set as 0, then those entries are required to be invalidated irrespective of the ASID value. The MMU-500 erroneously does not invalidate such entries.

Configurations affected

Configurations that enable Stage 1 Translation

Conditions

1. The MMU-500 fetches stage 1 page tables with nG bit set to 0. Typically, this occurs when the MMU-500 is sharing page tables with the processor and the software is using a Shared Virtual Memory scenario.

   Note: Linux does not use the SVM usage scenario, and so systems using Linux are not affected by this issue.

   Note: Systems using SVM Usage scenario in stage 2 translation are not affected by this issue.
2. The MMU-500 receives Invalidation transactions by Virtual Address that require the ASID to be matched:

   The following commands from the processor, on the DVM Interface:
   - TLBIVAE1IS.
   - TLBIVALE1IS.

The following commands are received on the register programming interface:
- SMMU_CBn_TLBIVA.
- SMMU_CBn_TLBIVAL.

Implications

This erratum can result in an incorrect translation being used for subsequent transactions that match the entry that has not been invalidated. This can result in Data integrity issues.

This erratum does not result in a Non-secure master being able to access Secure locations.

This erratum does not result in a master being able to bypass the protection that stage 2 translation provides.

General software workaround

If global stage 1 page table entries are not used, then this issue does not occur. Ensure that the stage 1 page tables that the MMU-500 accesses do not have global entries.

If this workaround is not possible, and global entries must be used, then use one of the following workarounds**.**

Software Workaround when only register programming interface is used for Invalidation

If global stage 1 page table entries are used, and when a register programming interface is used for invalidation, invalidation variants that use 'All ASID' must be used instead of a specific ASID invalidation:

- Instead of SMMU_CBn_TLBIVA, use SMMU_CBn_TLBIVAA.
- Instead of using SMMU_CBn_TLBIVAL, use SMMU_CBn_TLBIVAAL.

Because these invalidation commands are applied only on the specific context into which the invalidation is programmed, no over-invalidation occurs because of this, and so there are no performance implications.

Software Workaround when DVM Interface is used for invalidation

If Global Entries are used, then software can use register programming interface based Invalidation commands instead of using DVM Invalidation commands.

- Instead of TLBIVAE1IS from the processor, use SMMU_CBn_TLBIVAA in MMU-500 registers.
- Instead of TLBIVALE1IS from the processor, use SMMU_CBn_TLBIVAAL in MMU-500 registers.

Because global entries are not invalidated frequently, use the following DVM commands for invalidation:

- Instead of TLBIVAE1IS, use TLBIVAAE1IS or TLBIALLE1IS for invalidation.
- Instead of TLBIVALE1IS, use TLBIVAALE1IS or TLBIALLE1IS for invalidation.

Hardware integration workaround

In chips that have not been taped out, it is possible to add glue logic at the interface of the DVM bus input to the MMU-500 to modify the incoming DVM command to always have the 'ASID Match required' bit set to 0. However, this has a performance penalty of the DVM command invalidating all entries that match other conditions of the invalidation command with All ASIDs.

818688

SMMU_IDR0.BTM is 1 when BTM not supported by system

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2, r2p4.

Description

The SMMU_IDR0.BTM (Broadcast TLB Maintenance) field is intended to enable software to discover whether Inner Shareable TLB Invalidate system instructions that are executed on the CPU invalidate the MMU-500 TLB entries.

SMMU_IDR0.BTM is always 1, indicating that the MMU-500 supports Broadcast TLB Maintenance. However, for BTM to work, the MMU-500 must be integrated with an interconnect that supports AMBA Distributed Virtual Memory (DVM) operations. If the MMU-500 is integrated into a system without DVM support, then processor TLB Invalidate instructions cannot be used to invalidate MMU-500 TLB entries.

Conditions

1. The MMU-500 is integrated into a system without DVM support..
2. Software uses CPU TLB Invalidate instructions to invalidate MMU-500 TLB entries when SMMU_IDR0.BTM is 1.

Implications

Software cannot use SMMU_IDR0.BTM to determine whether processor TLB invalidate operations can be used to invalidate the MMU-500.

Note: Linux does not currently use processor TLB Invalidate instructions to invalidate MMU-500 TLB entries.

Software workaround

Software must obtain this information from elsewhere, for example, from Linux Device Tree.

837736

MMU-500 does not invalidate TTBR1 TLB entries in some conditions

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2, r2p4.

Description

The MMU-500 supports TLB maintenance operations through the DVM interface of the ACE-Lite port that the page table walker uses, and through the register programming interface.

When the MMU-500 receives an invalidation request by a Virtual Address (VA) that selects TTBR1 of a context, then the MMU-500 erroneously does not invalidate such entries.

Configurations affected

Configurations that enable Stage 1 Translation using AArch32 mode.

Conditions

1. The MMU-500 receives an input transaction with a VA that selects TTBR1 in a stage 1 context descriptor using AArch32 mode:
   - SMMU_CBA2Rn.TYPE is `0b01` or `0b11` for the context that is selected by the stream of the incoming transaction.
   - SMMU_CBA2Rn.VA64 is 0 for the context that is selected by the stream of the incoming transaction.
   - SMMU_CBARn.HYPC = 0 and SMMU_CBARn.MONC = 0.
   - SMMU_CBA2Rn.T0SZ, SMMU_CBA2Rn.T1SZ, Input VA [31:25] combination selects TTBR1 of the context descriptor.
2. The transaction completes page table walk and is cached in Macro-TLB of the TCU.
   - SMMU_CBn_ACTLR.CMTLB is 1 for the chosen context.
3. The MMU-500 receives Invalidation transactions by VA that match the TTBR1. Either:
   - One of the following commands is issued from the processor, on the DVM Interface.
     - TLBIVAE1IS.
     - TLBIVALE1IS.
   - One of the following commands is received on the MMU-500 register programming interface:
     - SMMU_CBn_TLBIVA.
     - SMMU_CBn_TLBIVAL.

Implications

This erratum can result in stage 1 translation being performed with stale page table entries, leading to data corruption.

This erratum does not result in a Non-secure master being able to access Secure locations.

This erratum does not result in a master being able to bypass the protection that is provided by stage 2 translation.

Software workaround

One of the following workarounds can be used**:**

1. If the register programming interface can be used to perform invalidations, in addition to issuing the TLB Invalidation by Virtual Address (SMMU_VBn_TLBIVA) that matches a TTBR1 entry with address bit 48 set to 0, issue an additional invalidation with address bit 48 set to 1 to invalidate the entry.
2. Issue TLBIASID from DVM, or SMMU_CBn_TLBIASID from register programming interface to invalidate the entry.
3. Issue TLBIALL from DVM, or SMMU_CBn_TLBIALL from register programming interface to invalidate the entry.

842870

Starvation of entries waiting in PTW Queue, when PTW Queue is full

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2.

Description

The Translation Control Unit (TCU) in MMU-500 performs Page Table Walks (PTW) to provide translation support for incoming transactions. The TCU supports multiple outstanding Page Table Walks in parallel. The number of outstanding Page Table Walks that can happen in parallel is configurable and varies between 4 and 32.

The TCU is designed to serve incoming transactions in a First Come First Served (FCFS) basis, when the Quality of Service (QOS) of the Translation Buffer Units (TBU) generating the transactions are the same. It achieves this by allocating the incoming entries as the last entry in a virtual FIFO, and shifting the entries up to 0, when a newer transaction arrives. The arbitration happens from 0 to the last entry in this virtual FIFO, thus achieving the FCFS arbitration. When all the FIFO Entries are occupied and one of the entries is evicted, the expected shift does not happen when a new entry is allocated to this FIFO, and the entry is written into the FIFO where the old entry was evicted. This results in the new transaction being given the priority of the evicted transaction. This results in the entries in the FIFO at a lower position than the newer transaction being given lower priority than expected, which could lead to those transactions being starved for long periods. The shifting happens only when there more is than one entry free in the FIFO, and a new transaction is being allocated. As long as the FIFO is being allocated a new transaction after one transaction is evicted, this condition persists.

Configurations affected

All Configurations.

Conditions

All Translation Queue entries in TCU are full.

Implications

When the erratum conditions occur, there is no functional impact, and there is no throughput impact. However, there could be latency impact on some transactions when the TCU is fully loaded.

In a typical system, this situation resolves itself as devices whose transactions at the head of the queue are being starved will stop pumping transactions resulting in the TCU becoming less loaded and the FCFS arbitration being applied again. However, it would be possible for a Direct Memory Access (DMA) device under malicious software control to create a Denial of Service situation by saturating the TCU and preventing the transactions stuck in the lower part of FIFO from being completed.

Workaround

There is no software workaround for this erratum.

1047329

Prefetch causes incorrect translation when both translation stages are enabled

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category B

Fault Status: Present in: r0p0, r1p0, r2p0, r2p1, r2p2, r2p4.

Description

Main TLB can be corrupted, when prefetch is enabled for the Stage 1 context of a combined Stage 1 and Stage 2 translation, and IPA2PA cache is enabled as well.

Configurations affected

All Configurations.

Conditions

1. A transaction received by a TBU results in a miss, and generates a translation request to the TCU.
2. The transaction requires combined Stage 1 and Stage 2 translation to be performed. (nested translation).
3. IPA2PA cache is enabled.
4. Main TLB is enabled for the Stage 1 context of that translation.
5. Prefetch is enabled for the Stage 1 context of that translation.
6. The lookup performed into IPA2PA cache on the prefetched Stage 1 Level 3 address receives a cache hit from the IPA2PA cache.
7. There is high load of updates happening to Main TLB from other translations, resulting in other microarchitectural conditions being met.

When all of the above conditions occur, the Main TLB might be updated with an entry for the original page which contains the translation result for the prefetched page.

Implications

Because an incorrect translation is cached in the Main TLB, future transactions for that context might be translated incorrectly.

Transactions will not access pages the context does not have permission to access.

Workaround

One of the following workarounds can be used:

1. IPA2PA cache can be disabled, by setting the SMMU_ACR.IPA2PA_CEN bit to 0.
2. Prefetch can be disabled for the Stage 1 context of the transactions that require nested translation. This can be done by setting the SMMU_CBn_ACTLR.CPRE bit to 0 for the Stage 1 context of the nested translation.

Main TLB can be disabled for the Stage 1 context of the transactions that require nested translation. This can be done by setting the SMMU_CBn_ACTLR.CMTLB bit to 0 for the Stage 1 context of the nested translation.

Category B (rare)

There are no errata in this category.

Category C

809669

Incorrect SMMU_CBn_FSYNR0.NSATTR logged for Secure contexts

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0. Fixed in r1p0.

Description

Fault Syndrome registers capture the input information regarding the faulted transaction for software debug purposes. The MMU-500 Stream-to-Context Register, SMMU_S2CRn, provides a pre-conditioning stage for attributes. The information that is recorded in the syndrome registers is post pre-conditioning stage.

For SSD Secure transactions, the SMMU_S2CRn pre-conditioning stage has an 'NSCFG' field, that modifies the NS-attribute, that is APROT[1], of the incoming transaction. This is reported to the 'SMMU_CBn_FSYNR0.NSATTR' field, if that transaction faults.

When the pre-conditioning stage indicates a Secure NS-attribute, and the Translation table stage NS-attribute is Non-secure, the reported NS-attribute is Non-secure instead of Secure because of this erratum.

Configurations affected

This erratum occurs only in RTL configurations that have been configured to support stage 1 translations in addition to stage 2 translations. That is, the RTL configuration option 'stage2 only translations' is set to false.

Conditions

This erratum occurs when all the following conditions occur:

- A Secure master initiates a transaction which maps to valid context of type 'Stage 1 context with stage 2 bypass', and the MMU for the context is enabled, by setting SMMU_CBn_SCTLR.M field to 1.
- SMMU_S2CRn.NSCFG = 2 indicating a Secure NS-attribute for the transaction, that is APROT[1] = 0, must be initiated, OR, SMMU_S2CRn.NSCFG = 0 indicating to not change the incoming NS-attribute and the incoming attribute is set to Secure, that is, APROT[1] = 0.
- The Secure attribute that is generated by the Translation Table walk stage indicates Non-secure.

Implications

This erratum does not affect the capability of software to detect the root cause for the fault because the fault model is not affected by the security state of the transaction, meaning that this erratum is benign.

Workaround

In the case that the SMMU_S2CRn.NSCFG field is programmed as Secure or Non-secure, then the software must use SMMU_CBFRSYNRAn.StreamID to find the SMMU_S2CRn that matched, and use the SMMU_S2CRn.NSCFG value instead of SMMU_CBn_FSYNR0.NSATTR.

In the case that the SMMU_S2CRn.NSCFG field is programmed as 'Use Incoming', the information about whether the pre-conditioned NS-attribute is Secure or not is not available, and there is no software workaround to work out the correct SMMU_CBn_FSYNR0.NSATTR.

809671

Allocate hint set incorrectly for some programming and input combinations

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0. Fixed in r1p0.

Description

The MMU-500 generates cache attributes as part of the translation process. The translation process consists of the incoming transaction being mapped to a Stream, and the Stream in turn being programmed to perform translation. The Stream to Context Register, SMMU_S2CRn, has a set of pre-conditioning attributes, to be used as a starting point for later stages of translation. In certain combinations of the SMMU_S2CRn register, the final allocate hint, that is generated as part of the cache attribute, is set when it is not supposed to be.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs when all the following conditions occur:

- S2CR.MTCFG=1 AND S2CR.MemAttr=?011x" or "1x01?, OR, S2CR.MTCFG=0 AND incoming cache is non-cacheable.
- S2CR.WACFG indicates 'Allocate'.
- WACFG from Translation Table Walk Stage indicates 'Use incoming allocate attribute'.

When the above conditions occur, the MMU-500 generates a cache attribute with 'Write-Allocate' hint set, when it is not supposed to be set.

The above erratum can also occur when the S2CR.RACFG fields are configured in a similar way.

A similar erratum occurs for the TRANSIENTCFG field. If SMMU_S2CRn.TRANSIENTCFG indicates a 'transient' transaction and later translation table attributes indicate 'use incoming', then the final generated attribute is 'transient' instead of non-transient.

Implications

When the erratum occurs, the downstream cache could allocate the transactions into the cache even if the allocation hints do not request this. This could lead to linefills and evictions that are not required. This erratum does not lead to data corruption, because the programmer cannot rely on allocation hints being obeyed.

Workaround

The S2CR.WACFG/RACFG/TRANSIENTCFG fields can be programmed to base values of non-allocate/non-transient. The later stage page tables can modify this attribute to a suitable value, instead of indicating 'use incoming'.

814019

Final stage 2 walk in a nested translation not cached in IPA2PA cache

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0. Fixed in r1p0.

Description

The MMU-500 supports stage 1 translation followed by stage 2 translation, that translates an input *Virtual Address* (VA) to an *Intermediate Physical Address* (IPA) and then translates that IPA to output a *Physical Address* (PA). The MMU-500 caches all the IPA to PA level walks in the IPA2PA cache, but does not cache the final IPA to PA walk in the IPA2PA cache.

Configurations affected

This erratum only occurs if the translation option chosen is nested, that is, stage 1 translation followed by stage 2 translation.

Conditions

This erratum occurs during page table walk of a stage 1 translation followed by stage 2 translation.

Implications

The final IPA to PA walk always results in external memory accesses. This results in loss of some performance. There is no functional impact.

Workaround

No workaround exists.

820470

Context fault register write can be prevented by transactions that continuously fault

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0.

Description

The MMU-500 has an AXI4 programming interface that is used for programming internal MMU-500 registers. The MMU-500 also has multiple *Translation Buffer Units* (TBUs) each of which contain a TLB and provide local translation support for the devices they are instantiated with. When a fault is detected for a transaction, the TBU sends the fault message to the TCU to be logged into the appropriate fault status register group. When the TBUs receive a continuous stream of transactions that are faulting, they generate a continuous stream of fault messages to the TCU. The programming interface does not accept any new transactions as long as the TCU is receiving a continuous stream of fault messages from the TBUs.

Configurations affected

This erratum affects the configurations where there are more than 15 TBUs.

Implications

The MMU500 AXI4 programming interface is stalled as long as all TBUs are receiving faulting transactions. This can lead to significant system delays.

Workaround

There is no workaround for the erratum.

820472

Bit 48 in Fault Address Register not updated in Stage 2 only configurations

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0.

Description

When address bit 48 is driven to `'b1` for a Stage 2 translation, the resultant fault that is logged has bit 48 in the SMMU_CBn_FAR register set to `'b0`, instead of `'b1`.

Configurations affected

This erratum occurs in configurations where the MMU-500 is configured as a Stage 2 only configuration.

Conditions

In Stage 2 only configurations, bit 48 in the address bus should be driven to 0. This erratum occurs if the system drives this bit to 1.

Implications

When this erratum occurs, software cannot diagnose the cause for the translation fault. There are no other functional issues.

Workaround

If there is no apparent reason for a translation fault when the MMU is enabled, or for an address size fault when the MMU is disabled, then the software can reason that the translation fault occurred because of bit 48 in the address being HIGH.

821769

Starvation of a TBU leads to system delays

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0, r1p0, r2p0. Fixed in r2p1.

Description

The MMU-500 has a *Translation Control Unit* (TCU) that provides address translation for multiple *Translation Buffer Units* (TBUs). A TCU can support up to 32 TBUs. In systems that contain 20 or more TBUs, if many those 20 TBUs have been programmed with a higher QoS than the other TBUs, and all the high priority TBUs are continuously generating page table walks, the page table walk that is generated by low priority TBUs is starved.

Configurations affected

This Erratum affects the configurations that have at least 20 TBUs.

Conditions

An invalidation command that is received by the MMU-500 on the programming and DVM Interface is broadcast from the TCU to the TBUs. The TCU must then wait for the invalidation acknowledge from all the TBUs before indicating that the invalidation is complete. The invalidation acknowledge generated by the TBUs with low QoS are not received by the TCU if it is stuck behind the page table walk request on the TBU-TCU channel. This leads to the invalidation command queue being built-up, and leads to the DVM and programming interfaces becoming backed up. This results in starvation of the DVM and/or programming channels.

Implications

When the erratum conditions occur, the DVM and/or Programming channel are delayed until the transactions from the higher priority TBUs are stopped.

Workaround

The Programming of QoS value per TBU should ensure that the number of TBUs that have a higher priority than a particular TBU are not more than 16.

821770

sACR.SMMU_PAGESIZE change results in a spurious update to the context bank0 register

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0, r1p0. Fixed in r2p0.

Description

The MMU-500 provides virtualization support through multiple context banks that can provide concurrent translation support for multiple page tables. The memory map for configuration the register space of each context bank is allocated such that they fall in separate pages in memory. The size of each page can be configured to be either 4K or 64K by setting the SMMU_SACR.PAGESIZE field. When this register is written to and the SMMU_SACR.PAGESIZE field is set to 1, the contents of the SMMU_CB0_TCR2 register are also updated with the same data as that written to the SMMU_SACR.

Configurations affected

This erratum affects all configurations.

Conditions

When the SMMU_SACR register is written to with the SMMU_SACR.PAGESIZE field set as '1'.

Implications

The contents of the SMMU_CB0_TCR2 register are corrupted when the erratum conditions occur.

Workaround

This issue can be worked around by reconfiguring the SMMU_CB0_TCR2 register after writing to the SMMU_SACR and whenever the SMMU_SACR.PAGESIZE field must be updated to '1'. Modification of the SMMU_SACR.PAGESIZE field should only occur during initialization.

825670

SMMU_IDR1.NUMS2CB is incorrect in Stage 2 only configurations

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r0p0, r1p0, r2p0. Fixed in r2p1.

Description

SMMU_IDR1.NUMS2CB denotes the number of context banks that support only Stage 2 translation. When the MMU-500 is configured to support only one Stage 2 translation, the number reflected in this field is `8'h00`, this is incorrect.

The value of `8'h00` is correct when the MMU-500 is configured to support all stages of translation, because there is no context that supports only Stage 2 because all contexts support both Stage 1 and Stage 2 translation in this mode.

Implications

This defect does not cause any functional issues.

Workaround

When the MMU-500 is configured to support stage 2 only translation, software must use the value of SMMU_IDR1.NUMCB to obtain the number of contexts supporting Stage 2 translation, instead of SMMU_IDR1.NUMS2CB.

829921

MAJOR version in IDR7 incorrect

Status

Affects: MMU-500 SMMU -System Memory Mgmt – TERM

Fault Type: Programmer, Category C

Fault Status: Present in: r1p0. Fixed in r2p0.

Description

The SMMU_IDR7.MAJOR_VERSION register field denotes the major revision of the MMU-500. The r1p0 version of the release should reflect this field as `'b1`, but it is incorrectly denoted as `1'b0`.

Implications

This erratum does not cause any functional implications.

Workaround

There are no functional issues, so no software workaround is required. The version of the MMU-500 can be uniquely identified as r1p0, by the following register reads:

- SMMU_IDR7 reads a value of `'h0`.
- PERIPHID4 reads a value `'h04`.

For versions higher than r1p0, the SMMU_IDR7 reads a correct value in the MAJOR_VERSION field. For the r0p0 version, the PERIPHD4 reads a value that is not equal to `'h04`.

MMU-500 r1p0 version corrects the following errata found in r0p0:

- 815719: Undetected permission fault when both stage 2 only and stage 1 followed by stage 2 translations are used.
- 815919: **ACREADY** depends on page table walk read completion, and this violates the AMBA spec.
- 817219: Incorrect memory attributes generation during certain page table walks.
- 809669: Incorrect SMMU_CBn_FSYNR0.NSATTR logged for Secure contexts.
- 809671: Allocate hint set incorrectly for some programming and input combinations.
- 814019: Final stage 2 walk in a nested translation that is not cached in IPA2PA cache.

Software can use the combination of PERIPHID4 and SMMU_IDR7 register reads to identify the MMU-500 r1p0 version, so that it is not necessary to apply workarounds that are specified for the above errata.

