ARM Processor

Cortex®-A72 MPCore (MP054)

Product Revision r1

Software Developers Errata Notice

Non-Confidential - Released

Non-Confidential Proprietary Notice

Web Address

http://www.arm.com

Feedback on content

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

- the document title
- the document number, ARM-EPM-012079
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Release Information

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 in Chapter 2. Fixed errata are not shown as updated unless the erratum text has changed. The summary table in section 2.2 identifies errata that have been fixed in each product revision.

14 Oct 2022: Changes in Document v13

Page	Status	ID	Cat	Rare	Summary of Erratum-
36	New	2660426	CatC		Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch

03 Nov 2021: Changes in Document v12

Page	Status	ID	Cat	Rare	Summary of Erratum-
35	New	2052435	CatC		An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update the debug registers

21 Feb 2020: Changes in Document v11

Page	Status	ID	Cat	Rare	Summary of Erratum-
31	Updated	1537003	CatC		Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort
34	New	1710472	CatC		Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort

20 Dec 2019: Changes in Document v10

Page	Status	ID	Cat	Rare	Summary of Erratum-
14	New	1655431	CatB		ELR recorded incorrectly on interrupt taken between cryptographic instructions in a sequence
33	New	1631484	CatC		Writes to normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode

20 Sep 2019: Changes in Document v9

Page	Status	ID	Cat	Rare	Summary of Erratum-
12	New	1319367	CatB		Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation
13	New	1387635	CatB		DSB is insufficient to ensure translation table entries being validated are visible to subsequent translations
29	New	1328359	CatC		Speculative TLB fills might occur past a DSB instruction
30	New	1406396	CatC		TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0
31	New	1537003	CatC		Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort

27 Jul 2018: Changes in Document v8

Page	Status	ID	Cat	Rare	Summary of Erratum-
11	Updated	859971	CatB	Rare	Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue
28	New	1185472	CatB		Exception packet for return stack match might return incorrect [E1:E0] field

31 Aug 2017: Changes in Document v7

Page	Status	ID	Cat	Rare	Summary of Erratum-
11	New	859971	CatB	Rare	Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue

06 May 2016: Changes in Document v6

Page	Status	ID	Cat	Rare	Summary of Erratum-
27	New	856026	CatC		Trace on packets from ETM are not generated in specific conditions around System Error exceptions

13 Jan 2016: Changes in Document v5

Page	Status	ID	Cat	Rare	Summary of Erratum-
8	New	853709	CatB		Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings
9	New	854173	CatB		Distributed Virtual Memory operations during hardware flush might cause deadlock

25	New	852124	CatC		Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions
26	New	854172	CatC		An external data snoop might cause data corruption when an Evict transaction is pending

01 Sep 2015: Changes in Document v4

Page	Status	ID	Cat	Rare	Summary of Erratum
22	New	851022	CatC		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores
23	New	852122	CatC		Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement
24	New	852123	CatC		Trace Context packet might not be output on a Reset or System Error exception

05 Jun 2015: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
19	New	848970	CatC		Data Synchronization Barrier might be stalled by a continuous stream of snoop transactions
20	New	850321	CatC		ATB stall from trace subsystem might deadlock the processor
21	New	850419	CatC		Cortex-A72 incorrectly allows access to GICv3 common registers in a specific configuration

19 Feb 2015: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
9	Updated	838569	CatA		DSB might be stalled by a steady stream of prefetch requests
17	New	842569	CatC		Accesses to RAMINDEX system control register might return incorrect data
18	New	843820	CatC		Syndrome value incorrect for software-induced Virtual Abort exception

03 Dec 2014: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
9	New	838569	CatA		DSB might be stalled by a steady stream of prefetch requests

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-A72 processors.

1.1. Scope of this document

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

- the current status of the defect
- where the implementation deviates from the specification and the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a 'work-around' where possible

This document describes errata that may impact anyone who is developing software that will run on implementations of this ARM product.

1.2. Categorization of errata

Errata recorded in this document are split into the following levels of severity:

Table 1 Categorization of errata

Errata Type	Definition
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.

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

2.2. Revisions Affected

Table 2 below lists the product revisions affected by each erratum. A cell marked with **X** indicates that the erratum affects the revision shown at the top of that column.

This document includes errata that affect revision r only.

Refer to the reference material supplied with your product to identify the revision of the IP.

Table 2 Revisions Affected

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r1p0
838569	CatA		DSB might be stalled by a steady stream of prefetch requests	X				
1655431	CatB		ELR recorded incorrectly on interrupt taken between cryptographic instructions in a sequence	X	X	X	X	X
1319367	CatB		Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	X	X	X	X	X
1387635	CatB		DSB is insufficient to ensure translation table entries being validate are visible to subsequent translations	X	X	X	X	X
854173	CatB		Distributed Virtual Memory operations during hardware flush might cause deadlock	X	X	X		
853709	CatB		Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings	X	X	X		
859971	CatB	Rare	Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue	X	X	X	X	
2660426	CatC		Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch	X	X	X	X	X
2052435	CatC		An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update debug registers	X	X	X	X	X
1710472	CatC		Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort	X	X	X	X	X
1631484	CatC		Writes to a normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode	X	X	X	X	X
1185472	CatC		Exception packet for return stack match might return incorrect [E1:E0] field	X	X	X	X	X

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r1p0
1328359	CatC		Speculative TLB fills might occur past a DSB instruction	X	X	X	X	X
1406396	CatC		TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0	X	X	X	X	X
1537003	CatC		Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort	X	X	X	X	X
856026	CatC		Trace on packets from ETM are not generated in specific conditions around System Error exceptions	X	X	X	X	X
854172	CatC		An external data snoop might cause data corruption when an Evict transaction is pending	X	X	X	X	X
852124	CatC		Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions	X	X	X		
852123	CatC		Trace Context packet might not be output on a Reset or System Error exception	X	X	X		
852122	CatC		Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement	X	X	X		
851022	CatC		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores	X	X	X	X	X
850419	CatC		Cortex-A72 incorrectly allows access to GICv3 common registers in a specific configuration	X	X			
850321	CatC		ATB stall from trace subsystem might deadlock the processor	X	X	X	X	X
848970	CatC		Data Synchronization Barrier might be stalled by a continuous stream of snoop transactions	X	X			
843820	CatC		Syndrome value incorrect for software-induced Virtual Abort exception	X				
842569	CatC		Accesses to RAMINDEX system control register might return incorrect data	X				

2.3. Category A

838569: DSB might be stalled by a steady stream of prefetch requests

Category A

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0

Description

A DSB following a TLB maintenance instruction executed on processor (PROC1) might be stalled by one or more other processors (PROCn) if the software executing on PROCn repeatedly activates the L1 hardware prefetcher. The L1 hardware prefetch requests must be presented to the L2 cache with specific timing and must win arbitration in a cluster of Cortex-A72 processors waiting to respond to the DSB completion request. If these prefetch requests occur repeatedly, with the correct timing, it is possible for the DSB completion to be stalled indefinitely.

Note: This erratum corresponds to issue #283 in the ARM internal tracking system.

Configurations affected

Systems with multiple processors.

Conditions

1) An ARM processor (PROC1) executes a TLB maintenance instruction followed by a DSB instruction.
2) One or more Cortex-A72 processors (PROCn) are running software that generates L1 hardware prefetches in a Cortex-A72 cluster that is trying to respond to the DSB completion request. Note that PROC1 and PROCn do not have to be in the same cluster.
3) The PROCn processors continuously execute instruction sequences that generate L1 hardware prefetch requests.
4) These prefetch requests arrive and win arbitration in the L2 cache before all existing outstanding prefetch operations have time to complete.

Implications

If the above conditions are met, the DSB instruction executing on PROC1 will be stalled until the continuous stream of prefetch requests stops.

Workaround

There is no workaround.

2.4. Category A (Rare)

There are no errata in this category.

2.5. Category B

853709: Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings

Category B

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2

Description

AArch32 memory domain access permissions are controlled by the DACR register. This register is accessible in AArch64 state by reading or writing the DACR32_EL2 register. If a Cortex-A72 processor attempts to write the DACR in AArch64 state using the DACR32_EL2 register, the updated register settings might not be seen by the processor.

Note: This erratum corresponds to issue #345 in the ARM internal tracking system.

Configurations affected

Systems running a hypervisor in EL2 in AArch64 state with AArch32 guests that use short-descriptor translation table format and domains to control access to pages.

Conditions

1) The hypervisor writes the DACR by writing to DACR32_EL2 before returning to an AArch32 guest.
2) The hypervisor does not write any other SPRs related to memory management, including SCTLR_EL1, TCR_EL1, TTBR0_EL1, TTBR1_EL1, or CONTEXTIDR_EL1 within the same bounds of context synchronizing events (such as an ISB).

Implications

If the above conditions are met, the Cortex-A72 processor might not properly use the updated DACR written by the hypervisor. This might result in spurious domain faults in the guest O/S, or possibly allow EL0 processes in the guest O/S access to the operating system privileged data or code.

Workaround

The hypervisor code in EL2 should write one or more of SCTLR_EL1, TCR_EL1, TTBR0_EL1, TTBR1_EL1, or CONTEXTIDR_EL1 after the write of DACR32_EL2.

854173: Distributed Virtual Memory operations during hardware flush might cause deadlock or data corruption

Category B

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2

Description

During execution of L2 hardware cache flush, as part of the powerdown sequence, Distributed Virtual Memory (DVM) operations received by the Cortex-A72 processor might be dropped leading to a deadlock or data corruption.

Note: This erratum corresponds to issue #358 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) L2 hardware cache flush is initiated following the powerdown sequence in the Cortex-A72 Technical Reference Manual by asserting L2FLUSHREQ.
2) During the L2 hardware cache flush sequence, the Cortex-A72 processor receives external DVM operations.
3) The L2 Fill Evict Queue (FEQ) becomes completely full with L2 hardware cache flush transactions or external DVM operations.
4) Very specific timing, so an L2 hardware cache flush transaction is allocated to a particular FEQ entry in the same cycle that another FEQ entry is deallocated.
5) DVM operation is later allocated to the last entry in the FEQ, causing the FEQ to be completely full.

Implications

If the above conditions are met, the Cortex-A72 processor drops the external DVM operation in step 5). This might cause data corruption for subsequent snoops and prevents the powerdown sequence from completing, because the Cortex-A72 processor does not respond to all the external DVM operations, and the system will deadlock.

Workaround

The workaround for this erratum applies only to the power management software, which in some cases is running on a system control processor (SCP), for the powering down of the Cluster and the L2 caches.

1) The preferred workaround involves the modification of the power management software running on the SCP using the L2 hardware cache flush mechanism as described in the ARM Cortex-A72 MPCore Processor Technical Reference Manual (rev0.2) section "Processor powerdown with system driven L2 flush". This mechanism involves using an implementation defined mechanism within the SoC to drive the signals L2FLUSHREQ and either ACINACTM or SINACT. In order to work around this erratum, while still using this mechanism, the power management software must be able to disable DVM operations before the SoC assertion of L2FLUSHREQ (step 6 of documented sequence) separately from disabling snoop requests (which is typically done by the management software between steps 8 and 9 of the documented sequence by writing to a register within the SoC level interconnect). For the ARM implemented interconnects, the following approaches are used:
    a. For CCI-400/500/550 based ACE interconnect systems, the power management software must program the Snoop Control Register (snoop_ctrl) to disable DVM operations to the Cortex-A72 processor and poll the Status Register to confirm changes to the Snoop Control Register have taken effect, just before step 6, where the SoC asserts L2FLUSHREQ. After step 8, where L2FLUSHDONE is asserted by the Cortex-A72 , program the Snoop Control Register to disable snoop requests and poll the Status Register to confirm changes have taken effect before asserting ACINACTM (this operation is required even without the errata workaround, but would also have including the disabling of DVM operations at this point).. Please refer to the appropriate CCI documentation for disabling snoops and DVM operations

b. For CCN-based AMBA 5 CHI interconnect systems, the power management software must write to the DVM Domain Control Clear Register (DDCR_Clear) to disable DVM operations to the Cortex-A72 processor and poll the DVM Domain Control Register (DDCR) to confirm changes to the DDCR have taken effect, just before step 6 where the SoC asserts L2FLUSHREQ. After step 8, where L2FLUSHDONE is asserted by the Cortex-A72, program the Snoop Domain Control Clear Register (SDCR_clear) to disable snoops and poll the Snoop Domain Control Register (SDCR) to confirm changes to the SDCR have taken effect (this operation is required even without the errata workaround but would also have including the disabling of DVMs at this point). Please refer to the appropriate CCN documentation for disabling snoops and DVM operations.

The steps required for power management operations are described in more detail in the ARM Power Control System Architecture v1.0 (ARM DEN 0050B).

2) If the SoC is such that DVM operations cannot be disabled independently from disabling snoops by power management software, then the caches must be cleaned and invalidated by software running the Cortex-A72 core as part of the power management software, following the steps in the Cortex-A72 MPCore Technical Reference Manual section “Processor powerdown without system driven L2 flush” using data or unified cache line clean and invalidate by set/way instructions (DCCISW) to clean and invalidate all data from the L2 data cache.

1319367: Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate incorrect translation

Category B

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

A speculative Address Translation (AT) instruction translates using registers associated with an out-of-context translation regime and caches the resulting translation in the L2 TLB. A subsequent translation request generated when the out-of-context translation regime is current uses the previous cached L2 TLB entry producing an incorrect virtual to physical mapping.

Note: This erratum corresponds to issue #411 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) A speculative AT instruction performs a table walk translating virtual address to physical address using registers associated with an out-of-context translation regime.
2) Address translation data generated during the walk is cached in the L2 TLB.
3) The out-of-context translation regime becomes current and a subsequent memory access is translated using previously cached address translation data in the L2 TLB, resulting in an incorrect virtual to physical mapping.

Implications

If the above conditions are met, the resulting translation would be incorrect.

Workaround

When context-switching the register state for an out-of-context translation regime, system software at EL2 or above must ensure that all intermediate states during the context-switch would report a level 0 translation fault in response to an AT instruction targeting the out-of-context translation regime. Note that a workaround is only required if the system software contains an AT instruction as part of an executable page.

1387635: DSB is insufficient to ensure translation table entries being validated are visible to subsequent translations

Category B

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

The Arm architecture states that a separate observer might observe a write to the translation tables at any time after the execution of the instruction that performed that write. However, it also states that the write is only guaranteed to be observable after the execution of a DSB instruction by the PE that executed the instruction that performed the write to the translation tables.

Because of this erratum, it is possible for a subsequent memory operation to generate a translation table walk, which might read a stale translation table descriptor before the write of the translation table descriptor being globally observed. This might lead to an unexpected Data Abort or incorrect translation.

Note: This erratum corresponds to issue #415 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) A store (ST1) writes a new valid mapping to the translation tables.
2) A DSB is executed, which is required by the architecture.
3) A subsequent memory operation executes which generates a translation table walk that uses the translation table entry written by (ST1), before (ST1) is globally observed.

Implications

If the above conditions are met, then the memory operation from Condition 3 might result in an unexpected Data Abort or an invalid translation leading to data corruption.

Workaround

Insert an ISB after the DSB to guarantee that the memory operation translates after the write has been globally observed, generating the proper translation.

1655431: ELR recorded incorrectly on interrupt taken between cryptographic instructions in a sequence

Category B

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

In aarch32 mode, if the code executed contains the sequence of cryptographic instructions:
AESE Qy, Qx
AESMC Qy, Qy

or

AESD Qy, Qx
AESIMC Qy, Qy

and an interrupt is asserted and taken just after execution of the first cryptographic instruction, then the ELR recorded as the return address might be incorrect leading to data corruption.

Note: This erratum corresponds to issue #423 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations which implement the cryptography extension.

Conditions

1) The core is in AArch32 state, either A32 or T32.
2) An instruction executes, producing a 32-bit result.
3) One of the two sequences of cryptographic instructions that are described above executes, using the 32-bit result from condition 2 as a source operand.
4) An interrupt is asserted and taken between the two instructions in the sequence.

Implications

If the above conditions are met, then return address that is recorded during the exception sequence might be incorrect, leading to data corruption by executing the second cryptographic instruction in the sequence twice.

Workaround

Arm expects the AES instructions are only used in hand optimised AES libraries. AES ECB and CBC modes are likely to load and store the vector registers in one operation. These are not affected by the erratum.

AES GCM mode may load a 32bit value to use as a counter. This fulfils condition 2. Implementations of AES GCM with a 32bit counter value can avoid condition 2 by copying the value to another Q register that is then used in the above sequence.

These crypto extensions are optional. An operating system may describe them as unimplemented when running aarch32 software.

2.6. Category B (Rare)

859971: Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue

Category B Rare

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3

Description

ARM architecture prohibits speculative instruction fetches to addresses that are marked as Execute-never (XN) in the page tables. However, where a Normal Write-Back Cacheable page with execute permission precedes a page with Execute-never (XN) permission in the virtual address space as configured by a combination of the stage1 and stage2 page tables, then the Cortex-A72 can issue a speculative instruction fetch to the first 64 bytes (cacheline) of the page with Execute-never (XN) permission. This speculative instruction fetch could result in a deadlock or introduce data integrity errors in the system by incorrectly accessing a read-sensitive device.

Note: This erratum corresponds to issue #378 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

5) A Normal Write-Back Cacheable page precedes an Execute-never (XN) page in the virtual address space.

6) Instruction cache prefetch is enabled.

Implications

When these conditions are met, a speculative instruction prefetch could access the read-sensitive device and cause a deadlock or introduce data integrity errors in the system.

Workaround

The Instruction prefetch can be disabled by writing CPUACTLR_EL1[32]=1. Alternatively, map read-sensitive devices to a location away from any instruction execution or by placing a Non-cacheable or Device empty page prior to the Execute-never (XN) page in the virtual address space.

2.7. Category C

842569: Accesses to RAMINDEX system control register might return incorrect data

Category C
Products Affected: Cortex-A72 MPCORE.
Present in: r0p0

Description

When reading the L1 instruction data array using the RAMINDEX register, the returned data should always be either 0x0, or the contents of a single RAM entry. However, if Debug state is entered while a table walk is occurring, and the L1 instruction data array is read before the table walk completes, the data returned might be incorrect.

Note: This erratum corresponds to issue #293 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) MMU is enabled.
2) An L1-I TLB miss occurs, causing a table walk.
3) Debug state is entered after the L1-I TLB miss but before the associated table walk completes.
4) A RAMINDEX operation is performed targeting the L1 instruction data array.

Implications

If the above conditions are met, the data read from the RAMINDEX operation might be incorrect.

Workaround

In Debug state, before writing the RAMINDEX register, disable the MMU by writing SCTLR_ELx.M = 0 where ELx is the exception level the RAMINDEX operation is performed at.

843820: Syndrome value incorrect for software-induced Virtual Abort exception

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0

Description

If a 32-bit hypervisor injects a virtual abort into a guest operating system by writing HCR.VA=1'b1, the DFSR syndrome value will be incorrect. The abort is correctly taken, but the DFSR will not provide correct information on what caused the asynchronous abort.

Note: This erratum corresponds to issue #297 in the ARM internal tracking system.

Configurations affected

Cortex-A72 systems with EL2 implemented and using AArch32.

Conditions

1) HCR.AMO is programmed to 1'b1, enabling virtual exceptions.
2) HCR.VA is programmed to 1'b1 to generate a virtual asynchronous abort.
3) An exception return is performed to Non-secure EL1.

Implications

If the above conditions occur, a virtual asynchronous abort will be correctly taken, but the DFSR syndrome value will be all zeroes, which is incorrect. This erratum is not expected to impact any existing systems. It requires a 32-bit hypervisor using virtual asynchronous aborts to generate exceptions in guest operating systems. There is no known hypervisor for Cortex-A72 that meets these conditions. Also, even if utilized, a virtual asynchronous abort of this type would be a fatal error, making the syndrome not helpful in system recovery.

Workaround

No workaround is required.

848970: Data Synchronization Barrier might be stalled by a continuous stream of snoop transactions

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1

Description

A core in a Cortex-A72 cluster executing a Data Synchronization Barrier (DSB) operation might be stalled by a continuous stream of snoop transactions to any core in the cluster. The Cortex-A72 waits for all outstanding snoop transactions in the cluster to finish before completing a DSB operation. If a pattern forms where new snoop transactions are continuously issued before existing snoop transactions complete, DSB completion will stall until there is a break in snoop activity.
Note: This erratum corresponds to issue #312 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) A core (CORE1) in a Cortex-A72 cluster executes a Data Synchronization Barrier (DSB) operation. An operation occurred since the last DSB that requires synchronization (that is, a TLB maintenance, IC invalidate, or brand predictor invalidate operation). If none occurred, the hardware optimizes the DSB and does not actually perform a synchronization.
2) A continuous stream of snoop transactions is processed by the shared L2 memory subsystem. These snoop requests can come from other cores in the cluster, or from outside the cluster on the external snoop request interface (AC Channel in AMBA 4 ACE configurations or RXSNP channel in AMBA 5 CHI configurations).
3) Specific latency and timing of snoop transactions such that there is always a new snoop transaction issued before existing snoop transactions complete.

Implications

If the above conditions are met, the DSB executing on CORE1 will stall until a period when all outstanding snoop transactions are complete and no new snoop transactions are pending. This erratum is not expected to impact typical systems, but malicious code could be written to exploit this in a denial of service attempt.

Workaround

A denial of service can be avoided if a timer-based interrupt source is used to interrupt all snoop generating masters periodically.

850321: ATB stall from trace subsystem might deadlock the processor

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3

Description

A processor can use the Wait for Event (WFE) or Wait for Interrupt (WFI) mechanism to enter low-power state. A processor can enter low-power state only after the Embedded Trace Macrocell (ETM) drains all trace bytes on the AMBA ATB interface. Under certain conditions an AMBA ATB stall can cause the processor to hang until the AMBA ATB stall condition is cleared. Some trace subsystems might require instructions to be executed on the processor to clear the AMBA ATB stall condition. An example of such trace subsystems involves draining the trace to the memory subsystem via an SMMU. A processor deadlock might occur when such trace subsystems are used.

Note: This erratum corresponds to issue #318 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) Trace subsystem requires instructions to be executed on the processor to clear an AMBA ATB stall condition.
2) The ETM is enabled.
3) The processor is executing a WFI or WFE instruction.
4) The ETM is unable to drain trace data as a trace stall is continuously asserted.

Implications

An interrupt might be raised in order to execute instructions on the processor to relieve the trace stall condition. This erratum means that the interrupt will not be taken and therefore a processor deadlock will occur.

Workaround

Ensure that the trace subsystem does not have interlock with software for draining the trace bytes.

850419: Cortex-A72 incorrectly allows access to GICv3 common registers in a specific configuration

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1

Description

If the Cortex-A72 internal GIC CPU interface is enabled and configured in system register mode, a specific access to one of the GICv3 common registers with ICH_HCR_EL2.TC=1 might incorrectly allow an access when an UNDEFINED exception or Monitor Trap should have been taken. ICH_HCR_EL2.TC=1 should not have any effect on GICv3 common register accesses in EL1S or EL2.

Note: This erratum corresponds to issue #319 in the ARM internal tracking system.

Configurations Affected

Systems implementing EL1, EL2 and EL3, and the external pin GICCDISABLE=0 (that is, Internal GIC CPU interface is enabled).

Conditions

1) Operating in System Register enabled mode, which has the system registers enabled for the current exception level.
2) Operating in exception level EL1S or EL2.
3) ICH_HCR_EL2.TC=1.
4) SCR_EL3.FIQ=1.
5) SCR_EL3.IRQ=1.
6) Read or write to one of the GICv3 common registers occurs. GICv3 common registers include: ICC_SGI0R_EL1, ICC_SGI1R_EL1, ICC_ASGI1R_EL1, ICC_CTLR_EL1, ICC_DIR_EL1, ICC_PMR_EL1, and ICC_RPR_EL1.

Implications

If EL3 is using AArch64, and the above conditions occur, a Monitor Trap exception should be taken because SCR_EL3.FIQ=1 and SCR_EL3.IRQ=1. If EL3 is using AArch32, an UNDEFINED exception should be taken. However, because of this erratum, the access is allowed.

Workaround

There is no workaround.

851022: Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, rop1, r0p2, r0p3

Description

In a coherent ACE system, a Write-Back No-Allocate (WBNA) store might be stalled if WriteUnique/WriteLineUnique (WU/WLU) transactions are enabled and the store is attempted when one or more cache evictions are pending. ACE requires that WU/WLU transactions do not bypass any outstanding evict type transactions (WriteBack/WriteEvict/WriteClean). To satisfy this requirement, a microarchitectural hazard is used to force a replay if a WU/WLU transaction is attempted when an eviction is pending. In rare scenarios with a persistent stream of L2 cache linefills and associated evictions, combined with significant backpressure in the interconnect, and with specific timing, it is possible for a WBNA store to be stalled indefinitely.

Note: This erratum corresponds to issue #327 in the ARM internal tracking system.

Configurations affected

Coherent ACE systems that enable WriteUnique/WriteLineUnique transactions.

Conditions

1) WriteUnique/WriteLineUnique transactions are enabled by changing the default value of L2ACTLR[4] and setting it to 1’b0.
2) A Cortex-A72 processor issues a Write-Back No-Allocate store (OP1). This can be a streaming store that was downgraded to Write-Back No-Allocate by the processor.
3) There is a pending eviction, forcing (OP1) to stall because of ACE requirements that require outstanding evictions to complete before WriteUnique/WriteLineUnique stores are performed.
4) A continuous stream of L2 cache linefills occurs, from other cores and/or prefetch, which triggers new evictions.
5) There is significant sustained backpressure in the interconnect, which keeps the system backed up and the ACE write channel queue near full.
6) Specific arbitration and timing conditions exist which, when combined with condition 5), trigger a microarchitectural hazard that causes condition 3) to repeat.

Implications

If the above conditions are met, (OP1) will stall until the specific timing conditions and backpressure in the L2 subsystem are relieved. Interrupts and barriers after the Write-Back No-Allocate store are also delayed until the store completes. The conditions for this erratum are rare and not expected to significantly impact real system performance. Additionally, most systems perform better when the reset value for L2ACTLR[4] is used and WriteUnique/WriteLineUnique transactions are disabled.

Workaround

If WriteUnique/WriteLineUnique transactions are not required, disable them by setting L2ACTLR[4] = 1'b1. This is the reset value. Otherwise, set L2ACTLR[7] = 1'b1 to enable L2 hazard detection timeout. This will force the L2 cache to periodically re-evaluate hazards, at which point the stall will be released.

852122: Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2

Description

The Embedded Trace Macrocell (ETMv4) architecture requires that when a trace flush is requested on the AMBA Trace Bus (ATB), a processor must complete any packets that are in the process of being encoded and output them prior to acknowledging the flush request. When trace is enabled, the Cortex-A72 processor attempts to combine multiple direct branch instructions into a single Atom packet. If a direct branch instruction is executed, and an Atom packet is in the process of being generated, Cortex-A72 does not force completion of the packet prior to acknowledging the flush request. This is a violation of the ETMv4 architecture.

Note: This erratum corresponds to issue #330 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) ETM is enabled.
2) Instruction tracing is active.
3) One or more direct branch instructions are executed.
4) An Atom packet is being encoded but is not complete.
5) A trace flush is requested on the AMBA ATB.

Implications

When the above conditions occur, the Atom packet being encoded should complete and be output prior to the trace flush request being acknowledged. Because of this erratum, the Atom packet is output after the flush is acknowledged. Therefore, it will appear to software monitoring the trace that the direct branch was executed after that requested flush.

Workaround

Enabling the timestamp by setting TRCCONFIGR.TS will solve the issue as it will complete the atom packets through the timestamp behavior.

852123: Trace Context packet might not be output on a Reset or System Error exception

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2

Description

In some scenarios, Cortex-A72 might not output a Context packet for the handler of the exception over the AMBA Trace Bus (ATB) when a reset exception or System Error exception causes an Exception level change. In all scenarios, the required Trace on, Exception, and Address trace packets are properly encoded and output.

Note: This erratum corresponds to issue #331 in the ARM internal tracking system.

Configurations Affected

All configurations are affected.

Conditions

The ETM is enabled but not currently active.
1) CID and VMID tracing is off (TRCCIDCCTLR0 is 0x0).
2) Reset exceptions are always traced (TRCVICTRL.TRCRESET = 1 or TRCVICTRL.TRCERROR = 1).
3) Exception level changes on a reset or System Error.
4) Tracing continues at handler.

Implications

If the above conditions occur, a Context packet is not output on AMBA ATB, as required by the architecture.
For the reset exception, software can assume that the Exception level will always change to EL3 coming out of reset.
A System Error might be routed to different Exception levels, so it would require software to query processor configuration registers bits to determine the Exception level for the context change.

Workaround

Enable CID and VMID tracing.

852124: Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2

Description

The processor might not output a Trace On packet in certain conditions surrounding Debug state. Specifically, a Trace On packet is not output when the processor generates a Debug Halt exception element that is traced followed by another exception element. The processor does generate the exception, address, and Context packets for this case.

Also, a Trace On packet is not output when the processor activates trace directly before a Debug Halt exception element wherein that debug entry is a P0 element that should be traced. The processor does generate the exception, address, and Context packets for this case.

Note: This erratum corresponds to issue #333 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

Condition Set A

1) ETM is enabled.
2) The core halts and a Debug Halt exception is traced.
3) a) An exception occurs on the first instruction after leaving Debug state.
4) b) A processor warm reset occurs while in Debug state.

Or

Condition Set B

1) ETM is enabled.
2) The core halts but the Debug Halt exception is not traced because tracing is inactive.
3) On exit from Debug state, one or more instructions are executed and are traced.
4) The core halts again, resulting in a Debug Halt exception being traced

Implications

If Condition Set A occurs, then a Trace On packet is not output after the Debug state exit, before the new exception is traced. The target address and context of the Debug state exit is traced correctly, as is the exception from step 3a or 3b. The missing Trace On element means that a trace analyzer might not highlight a gap in the trace, however this can be inferred by the trace analyzer because the Debug Halt exception is traced.

Or

If Condition Set B occurs, then a Trace On packet is not output before the first traced instruction out of Debug state at step 3. The target address and context are output correctly for the first instruction that is traced. The missing Trace On element means that a trace analyzer will not highlight the gap in the trace for the execution that was not traced before entering Debug state.

Workaround

For Condition Set A, a trace analyzer might be able to infer the Trace On packet because the Debug Halt exception is traced.

For both sets of conditions, a workaround is to disable and re-enable the ETM while the core is in Debug state.

854172: An external data snoop might cause data corruption when an Evict transaction is pending

Category C
Products Affected: Cortex-A72 MPCORE.
Present in: r0p0, r0p1, r0p2, r0p3

Description

If the Cortex-A72 is configured to send Evict transactions for a cache line in UniqueClean (UC) state, then the Cortex-A72 might return stale data on a snoop response.

Note: This erratum corresponds to issue #357 in the ARM internal tracking system.

Configurations affected

Systems using a non-ARM CHI-based interconnect that implement a snoop filter that can track cache lines with SharedClean (SC) state to the precise CPU cluster. CCN-5xx ARM implementations are not affected by this erratum.

Conditions

1) The Cortex-A72 is configured not to send data for UC evictions. This is accomplished by setting the value of L2ACTLR[14] to 1’b0. The default value of this bit is 1'b1.
2) The Cortex-A72 is configured to push dataless Evict transactions to the external world. This is accomplished by setting the value of L2ACTLR[3] to 1’b0. The default value of this bit is 1’b1.
3) The Cortex-A72 issues a dataless Evict transaction (one of SnpClean, SnpShared, and SnpUnique transactions) for a cache line in UC state.
4) The Cortex-A72 issues an Evict transaction and is waiting for a completion response (COMP) from the interconnect.
5) The interconnect issues a COMP followed by a snoop request to the same cache line address.
6) The Cortex-A72 receives the snoop request before the COMP because of a race condition in the interconnect.

Implications

If the above conditions are met, there is a possibility of data corruption.

Workaround

There are multiple workarounds:

1) Configure the Cortex-A72 to send data for UC evictions. This can be accomplished by setting L2ACTLR[14] to 1’b1.

Or

2) Configure the Cortex-A72 to not push Clean/Evict transactions to the external world. This can be accomplished by setting L2ACTLR[3] to 1’b1.

856026: Trace on packets from ETM are not generated in specific conditions around System Error exceptions

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3

Description

The processor might not output a Trace On packet in certain conditions surrounding System Error exception. Specifically, a Trace On packet is not output when the processor generates an asynchronous System Error exception element that is traced on a specific corner case while the trace is becoming active. The processor does generate the exception packet for this case.

Note: This erratum corresponds to issue #367in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) The ETM is enabled.
2) Trace becomes active through trace_force_active_excep_t2.
3) System Error exception packet asserted between the T3 and T4 stages in trace RTL.

Implications

A Trace On packet is not output for the exception packet. The exception packet is output correctly. The missing Trace On element means that a trace analyzer might not highlight the gap in the trace for the execution. Thus, a trace decompressor might incorrectly attribute a longer continuous stream. This should be a rare usage model, and there should be no serious effect if this erratum is encountered.

Workaround

Only trace exceptions. Such a workaround would require other features to be turned off in order to capture this issue.

1185472: Exception packet for return stack match might return incorrect [E1:E0] field

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0.

Description

When an abort or trap is taken at the target of an indirect branch matching the return stack value in the core ETM, an Exception packet might be generated with the 2-bit field [E1:E0] = 2'b10, which implies an Address element before the Exception element. When there is a trace return stack match, an Address element should not be generated before the Exception element. With [E1:E0] = 2'b10, the external Trace Analyzer might read the trace packet sequence to expect an Address element output before the Exception element and not complete the stack pop, which is incorrect. The correct value in the [E1:E0] field in the Exception packet for this case, should be 2'b01.

Configurations affected

All configurations are affected.

Conditions

1) ETM is enabled.
2) TRCCONFIGR.RS = 1'b1, which indicates the return stack is enabled.
3) Abort or trap is taken at the target of an indirect branch matching the return stack.

Implications

If the above conditions are met, then the external Trace Analyzer will not pop on the return stack match causing it to go out of sync with the core ETM.

Workaround

If tracing only EL0, then no workaround is required.
Otherwise, setting TRCCONFIGR.RS = 1'b0 to disable return stack is the workaround.

1328359: Speculative TLB fills might occur past a DSB instruction

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

In the whitepaper "Cache Speculation Side-channels" (https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability/download-the-whitepaper) issued by Arm in the response to the revelation of the "Spectre" side-channels, the claim is made that the combination of DSB SYS and ISB will prevent subsequent speculation. However, a single load, store, or other memory operation that makes a page translation that follows a DSB SYS + ISB can initiate a speculative table walk and fill a new TLB entry if the initial lookup results in a TLB miss before the completion of the DSB SYS + ISB. Correspondingly, the micro-architectural state of the processor can be affected by a speculative access from an instruction appearing after the DSB SYS + ISB.

Note: This erratum corresponds to issue #413 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) A DSB SYS + ISB precedes load or store instructions.

2) The address calculated by the load or store instruction misses the TLB.

3) The DSB SYS, ISB and load or store instructions are in the speculative path and ultimately flushed.

Implications

When these conditions are met, a single speculative access can alter the micro-architectural state of the TLB, so affecting the timing of subsequent accesses in a way that could reveal information about the address used for that speculative access.

The canonical "Spectre" variant 1 issue, that the DSB SYS + ISB can be used to avoid, involves a pair of speculative memory accesses in the shadow of a conditional branch that is sanitising an address offset. The first speculative memory access is used to access a secret selected by the attacker, and the second memory access uses this secret to form an address. The allocation of micro-architectural state based on this address depends on information in the secret, and by measuring the timing of subsequent accesses, information revealing the secret can be inferred.

Where the DSB SYS + ISB is placed before the first of these two speculative memory accesses, then the only effect of this erratum is that there may be a speculative page table walk using the address of the secret that has been supplied by the attacker. This cannot result in a revelation of the secret. Where the DSB SYS + ISB is placed before the second of these two speculative memory accesses, but not before the first, then the allocation in the TLB created by the second speculative memory access could reveal information about the secret retrieved speculatively by the first memory access.

Workaround

Where DSB SYS + ISB is used on this part to mitigate against the risk of Spectre variant 1, it should be placed before the first speculative memory access, not the second.

1406396: TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

TLBI instructions are not treating ASID[15:8] as zero when TCR_EL1.AS=0, as specified in the Arm Architecture Reference Manual. In this configuration, the bits are RES0, which should be written to zero by software, and ignored by hardware.

Note: This erratum corresponds to issue #416 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) TCR_EL1.AS=0.

2) A TLBI is executed with ASID[15:8] not equal to zero.

Implications

The TLBI will execute locally and broadcast with an ASID that is out of range for this configuration.

Workaround

This erratum can be avoided if software is properly writing zero to RES0 bits.

1537003: Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

Under a combination of conditions, a younger load re-using the same Group Identifier (GID) as an older device load, might report a synchronous external abort by improperly associating an external error detected by an older device memory access. Device loads resolve early and allow subsequent instruction execution to re-use the GID.

Note: This erratum corresponds to issue #418 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) Older device load executes and receives external error response from the interconnect, which schedules an asynchronous external abort.
2) 80 or more additional instructions execute which wrap the GID, causing a younger load to be assigned the same GID as the older device load.
3) The younger load has the same subset of the physical address PA[11:6] as the older device load.

Implications

When the above conditions occur, the younger load might incorrectly associate the external error for the older device load and generate a synchronous external abort, even when there is no abort that should be reported for the younger load. If this occurs, the asynchronous abort due to the external error on the older load remains pending until asynchronous aborts are unmasked.

There are two scenarios to consider to determine the implications of this behavior. First, the scenario where the spurious synchronous abort causes a change of exception level from lower privileged software to a higher privileged exception handler. Second, the scenario where the spurious synchronous abort is taken when executing higher privileged software and so the exception does not cause a change in exception level.

In the first scenario, it is expected that when higher privileged software takes a synchronous external abort from lower privileged software, and the cause cannot be determined, the higher privileged software will kill and start a new thread of execution at the lower level of privilege. This is possible because the synchronous external abort is precise and can always be attributed to the lower privileged software. When the new thread of execution is started, any pending asynchronous aborts will be taken once asynchronous aborts have been unmasked.

Therefore, under these circumstances the erratum might cause the lower privileged thread to be killed by the higher privileged software when the spurious synchronous abort is taken. However, the asynchronous abort due to the external error is still taken once aborts are unmasked and cause the lower privileged thread to be killed again and so the overall behavior of the system is not affected by the presence of the erratum.

In the second scenario, it is expected that when higher privileged software takes a synchronous external abort from the current exception level that this will either be considered fatal and the physical machine will need to be reset or it will cause the software to kill the higher privileged thread of execution and start a new thread. However, when higher privileged software takes an asynchronous abort from the current exception level it is possible for the higher privilege software to attribute the asynchronous external abort to lower privileged software. Upon entering the higher privilege exception level, the higher privilege code can associate any pending asynchronous aborts with the lower exception level by executing the following sequence:

```
DSB // ensure memory accesses have completed
MSR DAIFClr, #0b0100 // unmask SError
ISB // ensure any pending SError taken
```

Under these circumstances higher privileged software is able to attribute a pending asynchronous external abort to the lower privileged software and is therefore able to kill and start a new thread of execution at the lower privilege level. Any asynchronous aborts that are generated by memory accesses after this sequence has executed will be associated with the higher privileged software, and it is expected that this will be considered fatal and the physical machine will need to be reset.

This erratum means that it is possible for an external error that is generated by a device memory access executed by lower privileged software to cause a synchronous abort in higher privileged software that will cause the higher privileged thread to be killed or reset. Whereas the asynchronous abort that should be generated by the external error could be attributed to the lower privileged software and therefore only the lower privileged thread would need to be killed if the erratum had not occurred.

Workaround

When lower privileged software can generate device memory accesses that can respond with an external error response, and the higher level exception handler software attempts to attribute any pending asynchronous external aborts to the lower privileged software by executing the sequence:

```
DSB // ensure memory accesses have completed
MSR DAIFClr, #0b0100 // unmask SError
ISB // ensure any pending SError taken
```

Then the workaround for this Erratum is to ensure that this asynchronous abort attribution sequence must be executed before any explicit load instructions are executed in the exception handler. This is to ensure that any pending asynchronous aborts generated by the lower privileged software cannot cause a synchronous external abort to be taken by the higher privileged software.

1631484: Writes to normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

When the core is in write streaming mode, continuous writes of normal memory to the same 64 byte cacheline might prevent the writes from becoming globally observed in a finite amount of time.

Note: This erratum corresponds to issue #421 in the ARM internal tracking system.

Configurations affected

This erratum affects all multi-core configurations.

Conditions

4) A core is performing a series of writes that enable write streaming mode, gathering writes to consecutive 64-byte cachelines.
5) Once write streaming mode is enabled a sequence of writes targets the same 64-byte cacheline, but does not create any additional writes to any other cacheline.
6) Write streaming mode remains active and the writes to the current line being written are not made globally observed to other cores in the system.

Implications

When the above conditions are met, writes performed on the core in write streaming mode might not be made globally observed to other cores in the system in a finite amount of time that is required by the architecture.

Workaround

No workaround is expected for this erratum. This erratum is not expected to occur in realistic software, where a single cacheline is targeted by an infinite number of writes. In addition, there are several conditions that terminate the write streaming mode on the core performing the writes, for example, the execution of a memory barrier, the broadcast of DVM message followed by DSB, or the store streaming sequence being interrupted.

1710472: Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

Under an unusual combination of conditions, a synchronous external abort might be reported incorrectly for an older load instruction by falsely associating an external error from a younger load instruction. This can occur from normal memory accesses for aliasing conditions where two virtual addresses map to the same physical address, where the older load should report a permission or domain fault and the younger load detects the external error.

Note: This erratum corresponds to issue #424 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) Load A and Load B are to the same normal memory PA (but use different virtual addresses), where Load A appears in program order before Load B.
2) The translation tables are such that Load A gives a permission or domain fault on the translation table walk.
3) Load B is executed speculatively ahead of Load A and has an external error response.

Implications

When the above conditions occur, the external error response targeting Load B might be incorrectly associated with Load A and improperly reports a synchronous external abort for Load A instead of the permission or domain fault.

If this erratum is encountered, Arm does not expect it to have a significant impact, as any resolution of the permission or domain fault would cause Load A to receive an external abort. Portable software cannot rely on a permission or domain fault to prevent an external abort in these circumstances.

If the external abort is resolved and the program is restarted, condition 3 would not re-occur.

Workaround

There is no workaround.

2052435: An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update debug registers

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

When an MSR instruction and an APB write operation are processed on the same cycle, the MSR instruction might not update the destination register correctly.

Note: This erratum corresponds to issue #428 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs under the following conditions:

1) A CPU executes an MSR instruction to update any of following SPR registers:
   (a) DBGBCR<n>_EL1
   (b) DBGBVR<n>_EL1
   (c) DBGWCR<n>_EL1
   (d) DBGWVR<n>_EL1
   (e) OSECCR_EL1
2) An external debugger initiates an APB write operation for any of following registers:
   (a) DBGBCR<n>
   (b) DBGBVR<n>
   (c) DBGBXVR<n>
   (d) DBGWCR<n>
   (e) DBGWVR<n>
   (f) DBGWXVR<n>
   (g) EDECCR
   (h) EDITR
3) The SPR registers (for example, OSLSR_EL1.OSLK and EDSCR.TDA) and external pins are programmed to allow the following behavior:
   (a) The execution of an MSR instruction in condition 1 to update its destination register without neither a system trap nor a debug halt
   (b) The APB write operation in condition 2 to update its destination register without error
4) The MSR instruction execution in condition 1 and APB write operation in condition 2 happen in same cycle.
5) The MSR write and the APB write are to two different registers. The architecture specifies that it is the software or debugger's responsibility to ensure writes to the same register are updated as expected.

Implications

If the above conditions are met, an execution of the MSR instruction might not update the destination register correctly. The destination register might contain one of following values after execution:

1. The execution of the MSR instruction is ignored. The destination register of the MSR instruction holds an old value.
2. The execution of the MSR instruction writes an incorrect value to its destination register.

An external debugger and system software are expected to be coordinated to prevent conflict in these registers.

Workaround

No workaround is required for this erratum.

2660426: Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch

Category C

Products Affected: Cortex-A72 MPCORE.

Present in: r0p0, r0p1, r0p2, r0p3, r1p0. Open.

Description

A PE executing a PLDW or PRFM PST instruction that lies on a mispredicted branch path might cause a second PE executing a store exclusive to the same cache line address to fail continuously.

Note: This erratum corresponds to issue #432 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs under the following conditions:

1) One PE is executing store exclusive.
2) A second PE has branches that are consistently mispredicted.
3) The second PE instruction stream contains a PLDW or PRFM PST instruction on the mispredicted path that accesses the same cache line address as the store exclusive executed by the first PE.
4) PLDW/PRFM PST causes an invalidation of the first PE's caches and a loss of the exclusive monitor.

Implications

If the above conditions are met, the store exclusive instruction might continuously fail. It is not expected to be observed in real systems which have interrupts enabled, as the interrupt would break the pattern and enable the store exclusive to pass.

Workaround

No workaround is required for this erratum.