ARM Processor Cortex®-A53 MPCore

Product Revision r0

Software Developers Errata Notice

Non-Confidential - Released

Software Developers Errata 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-048406
- 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.

22 Oct 2019: Changes in Document v21

Page	Status	ID	Cat	Rare	Summary of Erratum
25	New	1530924	CatB		Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

03 Oct 2018: Changes in Document v20

Page	Status	ID	Cat	Rare	Summary of Erratum
58	Updated	855827	CatC		PMU counter values might be inaccurate when monitoring certain events

03 May 2017: Changes in Document v19

Page	Status	ID	Cat	Rare	Summary of Erratum
39	New	859081	CatB		Some AT instructions executed from EL3 might incorrectly report a domain fault
42	New	859372	CatB	Rare	Advanced SIMD integer multiply instructions in IT block might cause data corruption
62	New	859072	CatC		ETM might output an incorrect exception return address
63	New	859074	CatC		WFx can cause the PC to jump from lower VA subrange to upper VA subrange
64	New	859075	CatC		Accessing EDPCSR has side-effects when OS Lock is locked
65	New	859076	CatC		Instruction cache parity error might cause an incorrect abort to be taken
66	New	859077	CatC		Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR
67	New	859080	CatC		ETM trace reports incorrect branch target
68	New	859082	CatC		ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set
69	New	859083	CatC		Debug not entering Memory Access mode without setting EDSCR.ERR
70	New	859373	CatC		Mismatch between EDPRSR.SR and EDPRSR.R
71	New	859374	CatC		AArch64 memory streaming performance might be lower than expected when using a 64KB translation granule

18 Apr 2016: Changes in Document v18

Page	Status	ID	Cat	Rare	Summary of Erratum
36	New	853175	CatB		APB access to ETM space while core is in Retention will never complete
37	New	855871	CatB		ETM does not report IDLE state when disabled using OSLOCK
38	New	855873	CatB		An eviction might overtake a cache clean operation
40	New	855872	CatB	Rare	A Store-Exclusive instruction might pass when it should fail
50	Updated	836870	CatC		Non-allocating reads might prevent a store exclusive from passing
53	New	851672	CatC		ETM might trace an incorrect exception address
54	New	851871	CatC		ETM might lose counter events while entering wfx mode
55	New	852071	CatC		Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement
56	New	852521	CatC		A64 unconditional branch might jump to incorrect address
57	New	853172	CatC		ETM might assert AFREADY before all data has been output
58	New	855827	CatC		PMU counter values might be inaccurate when monitoring certain events
60	New	855829	CatC		Reads of PMEVCNTR<n> are not masked by HDCR.HPMN
61	New	855830	CatC		Loads of mismatched size might not be single-copy atomic

26 Mar 2015: Changes in Document v17

Page	Status	ID	Cat	Rare	Summary of Erratum
35	New	845719	CatB		A load might read incorrect data
52	New	845819	CatC		Instruction sequences containing AES instructions might produce incorrect results

20 Feb 2015: Changes in Document v16

Page	Status	ID	Cat	Rare	Summary of Erratum
22	New	843419	CatA		A load or store might access an incorrect address
34	New	843819	CatB		Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set

18 Nov 2014: Changes in Document v15

Page	Status	ID	Cat	Rare	Summary of Erratum
50	New	836870	CatC		Non-allocating reads might prevent a store exclusive from passing
51	New	836919	CatC		Write of JMCR in EL0 does not generate an UNDEFINED exception

08 Oct 2014: Changes in Document v14

Page	Status	ID	Cat	Rare	Summary of Erratum
20	New	835769	CatA		AArch64 multiply-accumulate instruction might produce incorrect result

29 Aug 2014: Changes in Document v13

No new or updated errata in this document version.

See the Summary Table for errata fixes in the latest product revision.

24 Jul 2014: Changes in Document v12

Page	Status	ID	Cat	Rare	Summary of Erratum
48	New	831920	CatC		HCR.DC being set when HCR.VM not set does not correctly affect Data Cache Invalidate by MVA/VA behaviour
49	New	831921	CatC		AuxReg field of ID_MMFR0 reports incorrect value

03 Jun 2014: Changes in Document v11

Page	Status	ID	Cat	Rare	Summary of Erratum
19	New	829070	CatA		Speculative data reads might be performed to Device memory

15 May 2014: Changes in Document v10

Page	Status	ID	Cat	Rare	Summary of Erratum
33	New	827319	CatB		Data cache clean instructions might cause overlapping transactions to the interconnect
31	New	826319	CatB		System might deadlock if a write cannot complete until read data is accepted
47	New	827971	CatC		Store release might not be multi-copy atomic

11 Mar 2014: Changes in Document v9

Page	Status	ID	Cat	Rare	Summary of Erratum
29	New	823819	CatB		A snoop request to a core coincident with retention entry might cause deadlock
30	New	824069	CatB		Cache line might not be marked as clean after a CleanShared snoop

27 Feb 2014: Changes in Document v8

Page	Status	ID	Cat	Rare	Summary of Erratum
24	New	823273	CatA	Rare	Load or store which fails condition code check might cause data corruption
46	New	822769	CatC		Mismatched aliases might cause deadlock

03 Feb 2014: Changes in Document v7

Page	Status	ID	Cat	Rare	Summary of Erratum
18	New	821523	CatA		Hazarding snoop request might cause deadlock

08 Jan 2014: Changes in Document v6

Page	Status	ID	Cat	Rare	Summary of Erratum
28	New	819472	CatB		Store exclusive instructions might cause data corruption

15 Nov 2013: Changes in Document v5

No new or updated errata in this document version.

See the Summary Table for errata fixes in the latest product revision.

05 Nov 2013: Changes in Document v4

Page	Status	ID	Cat	Rare	Summary of Erratum
27	New	814270	CatB		Misaligned PC and out-of-range address aborts might be taken to incorrect exception level
45	New	814271	CatC		CPUMERRSR register might be updated incorrectly after multiple ECC/parity errors

25 Sep 2013: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
17	New	812869	CatA		Instruction stream might be corrupted

16 Sep 2013: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
25	New	810919	CatB		Clearing of the global exclusive monitor can fail to set the event register
44	New	811819	CatC		A 64-bit load to Device memory which crosses a 64-byte boundary can access more data than is permitted

13 Aug 2013: Changes in Document v1

No new or updated errata in this document version.

See the Summary Table for errata fixes in the latest product revision.

Contents

Chapter 1. **Introduction**

This chapter introduces the errata notice for the ARM Cortex-A53 MPCore processor.

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 *rnpn* 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 r0 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	r0p4
843419	CatA		A load or store might access an incorrect address	X	X	X	X	X
835769	CatA		AArch64 multiply-accumulate instruction might produce incorrect result	X	X	X	X	X
829070	CatA		Speculative data reads might be performed to Device memory	X	X	X	X	
821523	CatA		Hazarding snoop request might cause deadlock	X	X	X		
812869	CatA		Instruction stream might be corrupted	X				
823273	CatA	Rare	Load or store which fails condition code check might cause data corruption	X	X	X		
1530924	CatB		Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	X	X	X	X	X
859081	CatB		Some AT instructions executed from EL3 might incorrectly report a domain fault	X	X	X	X	X
855873	CatB		An eviction might overtake a cache clean operation	X	X	X	X	X
855871	CatB		ETM does not report IDLE state when disabled using OSLOCK	X	X	X	X	X
853175	CatB		APB access to ETM space while core is in Retention will never complete	X	X	X	X	X
845719	CatB		A load might read incorrect data	X	X	X	X	X
843819	CatB		Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set	X	X	X	X	X
827319	CatB		Data cache clean instructions might cause overlapping transactions to the interconnect	X	X	X		
826319	CatB		System might deadlock if a write cannot complete until read data is accepted	X	X	X		

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r0p4
824069	CatB		Cache line might not be marked as clean after a CleanShared snoop	X	X	X		
823819	CatB		A snoop request to a core coincident with retention entry might cause deadlock	X	X	X		
819472	CatB		Store exclusive instructions might cause data corruption	X	X			
814270	CatB		Misaligned PC and out-of-range address aborts might be taken to incorrect exception level	X				
810919	CatB		Clearing of the global exclusive monitor can fail to set the event register	X				
859372	CatB	Rare	Advanced SIMD integer multiply instructions in IT block might cause data corruption	X	X	X	X	X
855872	CatB	Rare	A Store-Exclusive instruction might pass when it should fail	X	X	X	X	X
859374	CatC		AArch64 memory streaming performance might be lower than expected when using a 64KB translation granule	X	X	X	X	X
859373	CatC		Mismatch between EDPRSR.SR and EDPRSR.R	X	X	X	X	X
859083	CatC		Debug not entering Memory Access mode without setting EDSCR.ERR	X	X	X	X	X
859082	CatC		ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set	X	X	X	X	X
859080	CatC		ETM trace reports incorrect branch target	X	X	X	X	X
859077	CatC		Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR	X	X	X	X	X
859076	CatC		Instruction cache parity error might cause an incorrect abort to be taken	X	X	X	X	X
859075	CatC		Accessing EDPCSR has side-effects when OS Lock is locked	X	X	X	X	X
859074	CatC		WFx can cause the PC to jump from lower VA subrange to upper VA subrange	X	X	X	X	X
859072	CatC		ETM might output an incorrect exception return address	X	X	X	X	X
855830	CatC		Loads of mismatched size might not be single-copy atomic	X	X	X	X	X
855829	CatC		Reads of PMEVCNTR<n> are not masked by HDCR.HPMN	X	X	X	X	X
855827	CatC		PMU counter values might be inaccurate when monitoring certain events	X	X	X	X	X
853172	CatC		ETM might assert AFREADY before all data has been output	X	X	X	X	X
852521	CatC		A64 unconditional branch might jump to incorrect address	X	X	X	X	X
852071	CatC		Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement	X	X	X	X	X
851871	CatC		ETM might lose counter events while entering wfx mode	X	X	X	X	X
851672	CatC		ETM might trace an incorrect exception address	X	X	X	X	X
845819	CatC		Instruction sequences containing AES instructions might produce incorrect results	X	X	X	X	X
836919	CatC		Write of JMCR in EL0 does not generate an UNDEFINED exception	X	X	X	X	X
836870	CatC		Non-allocating reads might prevent a store exclusive from passing	X	X	X	X	X
831921	CatC		AuxReg field of ID_MMFR0 reports incorrect value	X	X	X	X	
831920	CatC		HCR.DC being set when HCR.VM not set does not correctly affect Data Cache Invalidate by MVA/VA behaviour	X	X	X	X	
827971	CatC		Store release might not be multi-copy atomic	X	X	X	X	
822769	CatC		Mismatched aliases might cause deadlock	X	X			
814271	CatC		CPUMERRSR register might be updated incorrectly after multiple ECC/parity errors	X				
811819	CatC		A 64-bit load to Device memory which crosses a 64-byte boundary can access more data than is permitted	X				

2.2.1. r0p0 implementation fixes

The following errata might be fixed in some implementations of r0p0. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[0]	810919	CatB	Clearing of the global exclusive monitor can fail to set the event register
REVIDR[1]	812869	CatA	Instruction stream might be corrupted

Note that there is no change to the MIDR which remains at r0p0 but the REVIDR is updated as above to indicate which errata are corrected. Software will identify this release through the combination of MIDR/MIDR_EL1 and REVIDR/REVIDR_EL1.

2.2.2. r0p1 implementation fixes

The following errata might be fixed in some implementations of r0p1. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[2]	821523	CatA	Hazarding snoop request might cause deadlock
REVIDR[6]	829070	CatA	Speculative data reads might be performed to Device memory

Note that there is no change to the MIDR which remains at r0p1 but the REVIDR is updated as above to indicate which errata are corrected. Software will identify this release through the combination of MIDR/MIDR_EL1 and REVIDR/REVIDR_EL1.

2.2.3. r0p2 implementation fixes

The following errata might be fixed in some implementations of r0p2. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[2]	821523	CatA	Hazarding snoop request might cause deadlock
REVIDR[4]	823273	CatA Rare	Load or store which fails condition code check might cause data corruption
REVIDR[5]	823819	CatB	A snoop request to a core coincident with retention entry might cause deadlock
REVIDR[6]	829070	CatA	Speculative data reads might be performed to Device memory
REVIDR[7]	835769	CatA	AArch64 multiply-accumulate instruction might produce incorrect result

Note that there is no change to the MIDR which remains at r0p2 but the REVIDR is updated as above to indicate which errata are corrected. Software will identify this release through the combination of MIDR/MIDR_EL1 and REVIDR/REVIDR_EL1.

2.2.4. r0p3 implementation fixes

The following errata might be fixed in some implementations of r0p3. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[6]	829070	CatA	Speculative data reads might be performed to Device memory
REVIDR[7]	835769	CatA	AArch64 multiply-accumulate instruction might produce incorrect result

Note that there is no change to the MIDR which remains at r0p3 but the REVIDR is updated as above to indicate which errata are corrected. Software will identify this release through the combination of MIDR/MIDR_EL1 and REVIDR/REVIDR_EL1.

2.2.5. r0p4 implementation fixes

The following errata might be fixed in some implementations of r0p4. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[7]	835769	CatA	AArch64 multiply-accumulate instruction might produce incorrect result
REVIDR[8]	843419	CatA	A load or store might access an incorrect address
REVIDR[9]	859374	CatC	AArch64 memory streaming performance might be lower than expected when using a 64KB translation granule

Note that there is no change to the MIDR which remains at r0p4 but the REVIDR is updated as above to indicate that the erratum is corrected. Software will identify this release through the combination of MIDR/MIDR_EL1 and REVIDR/REVIDR_EL1.

2.3. Category A

812869: Instruction stream might be corrupted

Category A
Products Affected: Cortex-A53 MPCore.
Present in: r0p0

Description

The Cortex-A53 MPCore processor includes a Branch Target Instruction Cache (BTIC). Under certain conditions after a BTIC hit the processor might execute some instructions using corrupted data.

Configurations Affected

All configurations.

Conditions

1) The MMU and the instruction cache must be enabled.
2) Instruction code included in a loop must contain:
    1. A branch of type: B, BL, CBZ/CBNZ, TBZ/TBNZ.
    2. The branch is at an address within 24 bytes of the end of a 4KB region (when mapped using a page size smaller than 1MB) or the end of a 1MB region (when mapped using a page size of 1MB or greater).
    3. There must be no unconditional branches between the target address of the branch and the next 8-byte boundary.
    4. The target address of the branch should not be within the last 8 bytes of a 64-byte cache line.
3) The loop must be executed again without any of the following conditions being met:
    1. An ISB or exception return being executed.
    2. An exception being taken.
    3. The fetch of an address which would result in a prefetch abort, breakpoint or vector catch. An instruction cache or TLB maintenance operation being executed.
    4. An address translation operation being executed.
    5. A change of security state.
    6. A write to the SCTLR, HSCTLR, SCR, HCR, HCR2, DACR, TTBR0, TTBR1, TTBCR, HTTBR, HTCR, VTTBR, VTCR, MAIR0, MAIR1, HMAIR0, HMAIR1, CONTEXTIDR, SCTLR_EL1, SCTLR_EL2, SCTLR_EL3, SCR_EL3, HCR_EL2, DACR32_EL2, TTBR0_EL1, TTBR1_EL1, TCR_EL1, TTBR0_EL2, TCR_EL2, VTTBR_EL2, VTCR_EL2, TTBR0_EL3, TCR_EL3, MAIR_EL1, MAIR_EL2, MAIR_EL3 or CONTEXTIDR_EL1.

If these conditions are met, then under unusual circumstances the data within the BTIC might be corrupted, causing incorrect data to be fetched between the first and second 8-byte address boundaries after the branch target address.

Implications

The Cortex-A53 processor could execute erroneous instructions, which would corrupt the flow of execution. There is no additional privilege/security exploit resulting from the presence of this erratum because the corrupted instruction will be executed in the current processor mode and security state (rather than some other errata-induced mode).

Workaround

No workaround is available.

821523: Hazarding snoop request might cause deadlock

Category A

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2

Description

The Cortex-A53 MPCore processor can accept snoop requests from an ACE interconnect. If these snoops match the address of an ongoing write request, then the hazarding logic will ensure that the write and the snoop are processed in the correct order. However, because of this erratum, in certain cases the snoop might not be processed correctly, which might result in deadlock.

Configurations Affected

This erratum only affects configurations of the Cortex-A53 processor with an AMBA 4 ACE master interface, when it can receive snoops from a coherent interconnect.

Conditions

1) The Cortex-A53 MPCore processor makes an external read request on the ACE interface.
2) The interconnect cannot accept the request, and so deasserts ARREADY.
3) There are at least six other requests in progress within the Cortex-A53 that are all also trying to arbitrate so that they can make a request on the ACE read channel.
4) There is a request, which starts after the six other requests, that needs to make a write request on the ACE interface. This write is an eviction from an L1 or L2 cache.
5) The interconnect makes a snoop request to the same address as the write request.

If these conditions are met then under certain timing conditions the Cortex-A53 processor might not respond to the snoop until after the interconnect has accepted the read requests, which could cause the system to deadlock.

Implications

If the erratum occurs, the processor might deadlock.

It is possible to meet the timing conditions for this erratum when Cortex-A53 is connected to CCI-400.

Workaround

There is no workaround for this erratum.

829070: Speculative data reads might be performed to Device memory

Category A

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3

Description

The v8 ARM architecture states that speculative data accesses are not permitted to any memory location with any Device memory attribute, with some exceptions. Because of this erratum, such speculative accesses might occur in non-exceptional cases.

Note: The ARMv7 and earlier architectures defined Device and Strongly-ordered memory types. Both of these are equivalent to forms of the Device memory type in ARMv8, and are affected by this erratum.

Configurations affected

Configurations of Cortex-A53 MPCore including the VFP + Neon Extension are affected.

Conditions

1) A load instruction is executed, accessing Device memory.
   - The load is not a floating-point or Advanced SIMD instruction.
2) Either:
   1. A floating-point, Advanced SIMD, or cryptography instruction is immediately before or immediately after the load instruction with no intervening exception, or
   2. There is a floating-point multiply-accumulate instruction (including Advanced SIMD floating-point) within the nine instructions preceding the load.
      - No exception or exception return is taken between the multiply-accumulate and the load.
      - There is no change of mode between the multiply-accumulate and the load.
      - There are no instructions reading the result of the multiply-accumulate or the floating-point status flags between the multiply-accumulate and the load.
      - There is no ISB instruction between the multiply-accumulate and the load.
      - For the purposes of this erratum, floating-point multiply-accumulate instructions are VFMA.F32, VFMA.F64, VFMS.F32, VFMS.F64, VFNMA.F32, VFNMA.F64, VFNMS.F32, VFNMS.F64, VMLA.F32, VMLA.F64, VMLS.F32, VMLS.F64, VNMLA.F32, VNMLA.F64, VNMLS.F32, VNMLS.F64, VRECPS.F32 and VRSQRTS.F32 in AArch32, and FMLA, FMLS, FMADD, FMSUB, FNMADD, FNMSUB, FRECPS and FRSQRTS in AArch64.

If these conditions are met then bytes within the same 16-byte aligned region as any of the accessed bytes might be speculatively read. The data from such a read will not be used by the processor.

Implications

If any peripherals are mapped using Device memory and have locations which have side effects on a read, such side effects could be observed for reads which were not requested by any load instructions executed by the processor. This erratum is classified as Category A as there is no workaround which can be generally applied to all systems.

Workaround

There is no generally viable workaround. The conditions of the erratum can be avoided by ensuring that any load instructions to Device memory have no adjacent floating-point, Advanced SIMD or cryptography instructions, and that the preceding nine instructions before a load to device memory do not contain any floating-point multiply-accumulate instructions. This can be simply achieved through inserting nine NOP instructions before and one after any such load.

835769: AArch64 multiply-accumulate instruction might produce incorrect result

Category A

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When executing in AArch64 state, some multiply-accumulate instructions which read an accumulator operand from the result of an earlier multiply instruction might produce incorrect results.

Configurations affected

All configurations of Cortex-A53 are affected.

Conditions

There are five similar sets of conditions, all only in AArch64 state, which can expose this erratum.

In the following conditions:

- "A multiply instruction" is defined as one of MADD, MSUB, MUL, SMADDL, SMSUBL, SMULH, SMULL, UMADDL, UMSUBL, UMULH or UMULL.
- "A multiply-accumulate instruction" is defined as one of MADD, MSUB, SMADDL, SMSUBL, UMADDL or UMSUBL.
- "A 64-bit multiply-accumulate instruction" is defined as a multiply-accumulate instruction which has the 'sf' field (bit 31) set in the instruction encoding.

Condition 1:

1) The following two instructions are executed in either order:
   - A load or store instruction using pre-indexed writeback.
   - A multiply instruction.
2) A load, store or prefetch instruction using the same base register (Rn) as the earlier load or store is executed.
3) A 64-bit multiply-accumulate instruction is executed, which uses the result of the earlier multiply instruction as the accumulator register (Ra). This instruction cannot read any register written by the instruction in subcondition 2.
4) Between these four instructions there must be no other instructions.

Condition 2:

1) The following two instructions are executed in either order:
   - An ADRP instruction.
   - A multiply instruction.
2) A load, store or prefetch instruction using the destination register (Rd) of the ADRP as the base register (Rn) is executed.
3) A 64-bit multiply-accumulate instruction is executed, which uses the result of the earlier multiply instruction as the accumulator register (Ra). This instruction cannot read any register written by the instruction in subcondition 2.
4) Between these four instructions there must be no other instructions.

Condition 3:

1) An MSR instruction writing to SP_EL0, SP_EL1, SP_EL2, ELR_EL1, ELR_EL2 or ELR_EL3 is executed.
2) A multiply instruction is executed.
3) A load, store or prefetch instruction is executed.
4) A 64-bit multiply-accumulate instruction is executed, which uses the result of the earlier multiply instruction as the accumulator register (Ra). This instruction cannot read any register written by the instruction in subcondition 3.
5) There can be up to one additional instruction inserted in this sequence either before or after the multiply in subcondition 2, which could be a branch instruction.

Condition 4:

1) The following two instructions are executed in either order:
   - Any operation writing the stack pointer.
   - A multiply instruction.
2) A load, store or prefetch instruction using the zero register as an address offset register (Rm) is executed.
3) A 64-bit multiply-accumulate instruction is executed, which uses the result of the earlier multiply instruction as the accumulator register (Ra). This instruction cannot read any register written by the instruction in subcondition 2.
4) Between these four instructions there must be no other instructions.

Condition 5:

1) An instruction is executed which writes to an integer register, referred to as A. This can be:
   - A single-register load, where A is the transfer register (Rt).
   - An LDP or LDNP of 64-bit integer registers, where A is the second transfer register (Rt2).
   - An integer multiply or divide, where A is destination register (Rd).
   - An FCVT*, FMOV, UMOV or SMOV, where A is destination register (Rd).
2) An integer data processing instruction is executed, which writes to A but does not read A.
3) A multiply instruction is executed.
4) A load, store or prefetch instruction is executed using A as the base register (Rn) or as the offset register (Rm) if using register-offset addressing.
5) A 64-bit multiply-accumulate instruction is executed, which uses the result of the earlier multiply instruction as the accumulator register (Ra). This instruction cannot read any register written by the instruction in subcondition 4.
6) There can be up to one additional instruction inserted in this sequence either before or after the multiply in subcondition 3, which could be a branch instruction.

Implications

If any of these sets of conditions are met, then the upper 32 bits of the result of the final multiply-accumulate instruction might be computed incorrectly.

Workaround

The only viable workaround is to avoid any of these code sequences, typically by avoiding the use of multiply-accumulate instructions, or by inserting a NOP between any adjacent load/store/prefetch instruction and multiply-accumulate instruction with no data dependency between them.

843419: A load or store might access an incorrect address

Category A

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When executing in AArch64 state, a load or store instruction which uses the result of an ADRP instruction as a base register, or which uses a base register written by an instruction immediately after an ADRP to the same register, might access an incorrect address.

Configurations affected

All configurations of Cortex-A53 are affected.

Conditions

There are two instruction sequences which can trigger this erratum, both only in AArch64 state.

Sequence 1:

1) An ADRP instruction, which writes to a register Rn.
   - This instruction must be located in memory at an address where the bottom 12 bits are equal to 0xFF8 or 0xFFC.
2) A load or store instruction:
   - This can be:
     - A single register load or store, of either integer or vector registers
     - Or an STP or STNP, of either integer or vector registers
     - Or an Advanced SIMD ST1 store instruction.
   - This must not write to Rn.
   - This may optionally read Rn (either as an address register or a data register to be stored).
   - If a load, this must access a region of memory which crosses a 64-bit boundary.
   - If a store, this must access a region of memory which crosses a 128-bit boundary.
3) There can optionally be one instruction after instruction 2.
   - This cannot be a branch.
   - This cannot write Rn.
   - This may optionally read Rn.
4) A load or store instruction from the "Load/store register (unsigned immediate)" encoding class, using Rn as the base address register.

Sequence 2:

1) An ADRP instruction, which writes to a register Rn.
2) Another instruction which writes to Rn.
   - This cannot be a branch or an ADRP.
   - This cannot read Rn.
3) Another instruction.
   - This cannot be a branch.
   - This cannot write Rn.
   - This may optionally read Rn.
4) A load or store instruction from the "Load/store register (unsigned immediate)" encoding class, using Rn as the base address register.

Implications

If the above conditions are met, then the load or store instruction at position 4 in either sequence might access an incorrect address. In all cases, the address accessed will undergo the same permission checks and translation as if that were the correct address for the instruction. In no case will the load or store access memory which the current exception level does not have access permissions for.

Workaround

The only viable workaround for this erratum is to avoid the triggering code sequences. Typically this would be achieved by:

- For sequence 1, ensuring that any loads or stores using the result of an ADRP are immediately after the ADRP when the instructions are in different 4KB regions, or by inserting NOP instructions before the final load or store to ensure there are more than two instructions between the ADRP and the load or store.
- For sequence 2, ensuring that the result of an ADRP is not overwritten by a consecutive instruction which does not read that result, by either removing the ADRP or placing a NOP after the ADRP.

2.4. Category A (Rare)

823273: Load or store which fails condition code check might cause data corruption

Category A Rare
Products Affected: Cortex-A53 MPCore.
Present in: r0p0, r0p1, r0p2

Description

Under rare circumstances a conditional load instruction which fails its condition code check might cause data corruption or deadlock.

Configurations Affected

This erratum affects all configurations of Cortex-A53.

Conditions

1) The core is executing in AArch32.
2) A load instruction is executed that satisfies the following requirements:
   - The instruction is conditional.
   - The instruction fails its condition code check.
   - The data addressed by the instruction given correct register inputs crosses an 8-byte boundary (including cases where an alignment fault would have been generated if the instruction had passed its condition code check).
3) Within the six instructions before the load instruction is an instruction which writes to a register used in the address calculation of the load.
   - This instruction must have the opposite condition code to the load instruction.
   - There must be no instructions which set the condition code flags between this instruction and the load instruction.
4) The load instruction must stall in the pipeline due to one of the following reasons:
   - A VFP divide or square root instruction completes at the same time the load is issued.
   - The ETM is enabled and requests a pipeline stall. ETM stalling is enabled by setting TRCSTALLCTLR.ISTALL.
   - FP/Neon retention is enabled by setting the CPUECTLR.FPRETCTL field to a non-zero value, and the load is either a VFP/Neon load or is adjacent to a VFP or Neon instruction.
   - The load instruction is adjacent to a conditional integer divide instruction, which stalls due to an earlier integer divide which has not yet completed.
   - The load is to the program counter, or the instruction following the load is a BLX (register) instruction.

If these conditions are met, then under rare conditions data memory might be corrupted.

Implications

It is believed that the various conditions required to cause this erratum are very unlikely to occur together in practice. Defect 823274 on the Cortex-A7 MPCore processor shares the same root cause, and has not been observed in deployed systems. As such, the practical implications for real systems should be minimal.

Workaround

There is no complete workaround for this erratum. Some of the stall conditions can be avoided by disabling ETM stalling in the TRCSTALLCTLR and disabling FP/Neon retention in the CPUECTLR.

2.5. Category B

1530924: Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4 Open

Description

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

Configurations Affected

All configurations are affected.

Conditions

1) A speculative AT instruction performs a table walk, translating a virtual address to a physical address using registers associated with an out-of-context translation regime.
2) Address translation data that is generated during the walk is cached in the 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 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. A workaround is only required if the system software contains an AT instruction as part of an executable page.

810919: Clearing of the global exclusive monitor can fail to set the event register

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0

Description

The Cortex-A53 MPCore processor implements the ARMv8-A architecture. One of the features of this architecture is that if the global exclusive monitor for a processor changes from the Exclusive state to the Open state, the event register for that processor is set.

In some circumstances on the Cortex-A53 processor this event might not be generated, meaning that the processor will not exit standby state.

Configurations Affected

All configurations.

Conditions

1) A load-exclusive operation is executed to an address marked as cacheable and shareable. Concurrently with (1), another agent in the system (another core in the cluster, or another master connected using a coherent interconnect) accesses the same address, causing the cache line to be evicted from the processor's cache.
2) The processor executes a WFE instruction.

If the above conditions are met then the WFE instruction can cause entry into standby state rather than resuming execution after the instruction.

Implications

This erratum can cause software relying on this behavior to hang. This has been observed in spinlock implementations within OS kernels.

Workaround

The effects of this erratum can be mitigated using the event stream feature of the Generic Timer. Using CNTKCTL/CNTKCTL_EL1 or CNTHCTL/CNTHCTL_EL2, the event stream can be configured to periodically generate an event on the processor, which would wake up a processor which had hung due to this erratum and allow software to continue execution.

814270: Misaligned PC and out-of-range address aborts might be taken to incorrect exception level

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0

Description

The ARMv8 architecture requires that:

- Branches to unaligned addresses cause an instruction fault.
- Accesses in AArch64 to virtual addresses outside a 48-bit range from the appropriate base cause a translation fault.

Because of this erratum such faults might be reported at EL2 rather than at EL1.

Configurations Affected

All configurations are affected.

Conditions

- Execution is at the non-secure EL0 or EL1 exception levels, in either AArch32 or AArch64.
- A branch is executed to an address which is either:
  - Unaligned (that is, not 4-byte aligned for the A32 and A64 instruction sets or not 2-byte aligned for the T32 instruction set).
  - Not within the ranges 0x0000000000000000-0x0000FFFFFFFFFFFF or 0xFFFF000000000000-0xFFFFFFFFFFFFFFFF.
- The memory location addressed by the (sign-extended) bottom 49 bits of the branch target would cause a stage 2 permission fault if accessed from the current exception level.

If these conditions are met, then the processor might incorrectly take the exception caused by the instruction fault at EL2 (in EL2H or Hyp) mode, rather than at EL1.

Implications

Some fault types might be reported to the hypervisor rather than the operating system.

Workaround

Hypervisor software, when handling an instruction fault from a lower exception level, can identify these cases from examination of the HFAR/FAR_EL2. The hypervisor can then emulate the correct exception and pass the exception to the operating system fault handler.

819472: Store exclusive instructions might cause data corruption

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1

Description

If a Cortex-A53 processor is executing a load and store exclusive sequence at the same time as a processor in another cluster is executing a cache maintenance operation to the same address, then this erratum might cause data corruption.

Configurations Affected

This erratum affects all configurations of the Cortex-A53 processor with an L2 cache present when it is connected to a coherent interconnect.

Conditions

1) A load exclusive instruction is executed to shareable, cacheable memory.
2) A store exclusive instruction is executed to the same address.
3) The store exclusive instruction passes the exclusive monitor.
4) The cache line is in a SharedDirty state in the Cortex-A53 cluster, and is present in L2 and in the L1 cache of the processor performing the store exclusive.
5) An external CleanShared snoop for the same cache line address is sent to the Cortex-A53 cluster while the store exclusive is executing. This snoop type is typically generated from a cache maintenance operation.

Implications

If the erratum occurs, a subsequent load to the same address might not observe the data from the store exclusive or other stores from that processor, resulting in data corruption.

In a system that only contains clusters of Cortex-A53 or Cortex-A57 processors this will only occur if one processor is executing a DCCMVAC or DCCMVAU instruction to the same address to which the exclusive access is targeted.

Workaround

Software that uses DCCMVAC or DCCMVAU instructions can replace them with DCCIMVAC instructions to avoid this erratum.

Note that this workaround must be applied to all the masters in the system that might generate a snoop into the Cortex-A53 cluster. For example, on a big.LITTLE configuration with one Cortex-A53 processor and one Cortex-A57 processor, the workaround needs to be applied to all software running on the Cortex-A57 processor.

On a Cortex-A57 processor it is possible to set bit 44 in the CPUACTLR_EL1 register to automatically convert all DCCMVAC and DCCMVAU instructions into DCCIMVAC instructions.

823819: A snoop request to a core coincident with retention entry might cause deadlock

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2

Description

The Cortex-A53 MPCore processor supports a dynamic retention feature, where a core can execute a WFI or WFE instruction and subsequently enter a low-power retention state. Because of this erratum, under some conditions the processor might deadlock on core retention entry.

Configurations affected

This erratum affects configurations of Cortex-A53 that make use of the CPU retention feature.

Conditions

1) Core retention is enabled by setting CPUACTLR.CPURETCTL to a non-zero value.
2) The core executes a WFI or WFE instruction, entering into standby mode.
3) The timer for retention entry expires, and the processor takes CPUQACTIVE for that core low.
4) The power controller takes CPUQREQn for that core low, requesting entry into retention state.
5) Coincident with the request from the power controller, an access is made which requires a snoop to be sent to the core in standby. This could be caused by:
   - An access to an address present within the L1 data cache of the core from either another core in the processor or from an agent outside the processor received over a coherent interconnect.
   - An instruction cache or TLB maintenance operation executed by another core within the processor.
   - A DSB instruction executed by another core within the processor.
   - A DVM message transaction being sent to the processor from a coherent interconnect.

If these conditions are met, then the processor might deadlock.

Implications

In a system implementing CPU retention where one core can enter retention dynamically while other cores or processors are executing, deadlock conditions might be observed. If CPU retention is disabled, this erratum will not occur, but overall system power consumption might be higher.

Workaround

This erratum can be avoided by disabling CPU retention, by setting CPUACTLR.CPURETCTL to zero.

824069: Cache line might not be marked as clean after a CleanShared snoop

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2

Description

If a Cortex-A53 processor is executing a store or PLDW instruction at the same time as a processor in another cluster is executing a cache maintenance operation to the same address, then this erratum might cause a clean cache line to be incorrectly marked as dirty.

Configurations affected

This erratum affects all configurations of the Cortex-A53 processor when it is connected to a coherent interconnect.

Conditions

1) A store or PLDW instruction is executed to shareable cacheable memory, or the hardware data prefetcher spots a pattern of stores and starts prefetching a cache line.
2) The cache line containing that address is in a SharedDirty state in the Cortex-A53 cluster, but is not present in the L1 cache of the CPU executing the store or PLDW.
3) An external CleanShared snoop for the same cache line address is sent to the Cortex-A53 cluster while the store or PLDW is executing. This snoop type is typically generated from a cache maintenance operation. The snoop gets a PassDirty response indicating that the responsibility for the dirty data is being passed to the interconnect.
4) Either:

- The line is fetched by the PLDW or data prefetcher but the CPU does not perform a store to that address. The line might be marked as dirty even though there was no store performed.
- A second external snoop for the same cache line is sent to the Cortex-A53 cluster while the store or PLDW is still executing. This snoop might get a dirty response when it should see clean data.

Implications

In a system that only contains clusters of Cortex-A53 or Cortex-A57 processors, a CleanShared snoop will only occur if one processor is executing a DCCMVAC or DCCMVAU instruction.

If the erratum occurs, the cache line could be marked as dirty in more than one cluster. This violates the coherency protocol, however for many coherent interconnects it will not cause any functional failure. It will not cause any failure with CCI-400 or CCN-504.

The erratum could also cause the line to remain incorrectly marked as dirty after a DCCMVAC or DCCMVAU instruction. This could then cause the line to be written back to memory when it did not need to be. This will not cause any functional failure in the majority of systems, but could be visible if that address was being used with software cache maintenance to communicate with a non-coherent master.

Workaround

The majority of systems will not need a workaround for this erratum. For any systems that do need a workaround, any software that uses DCCMVAC or DCCMVAU instructions can replace them with DCCIMVAC instructions, or their AArch64 equivalents.

Note that this workaround must be applied to all the masters in the system that might generate a snoop into the Cortex-A53 cluster. For example, on a big.LITTLE configuration with one Cortex-A53 processor and one Cortex-A57 processor, the workaround needs to be applied to all software running on the Cortex-A57 processor.

On a Cortex-A57 processor it is possible to set bit 44 in the CPUACTLR_EL1 register to automatically convert all DCCMVAC and DCCMVAU instructions into DCCIMVAC instructions.

826319: System might deadlock if a write cannot complete until read data is accepted

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2

Description

The Cortex-A53 MPCore processor supports access to other processors and peripherals through an AMBA AXI4 ACE interface. It is intended that the processor always accepts read data presented on the read data channel in a timely manner. It should not require another transaction to complete before it accepts the read data. Because of this erratum, the processor might deassert RREADYM until a write that is presented on the write channel is accepted by the interconnect and receives a response from the interconnect. If the system is unable to accept the write until the processor has accepted read data then RREADYM can remain deasserted indefinitely and the system can deadlock.

Configurations Affected

This erratum only affects configurations of the Cortex-A53 processor with an AMBA 4 ACE or AXI master interface and an L2 cache.

To be affected by this erratum, the system that Cortex-A53 is connected to must also contain a peripheral or other component that contains a dependency between the read and write channels. The dependency must prevent the peripheral from accepting or responding to a write until it finishes processing a read.

Conditions

1) A DCCMVAC, DCCMVAU, or DCCSW instruction (or the equivalent DC CVAC, DC CVAU, or DC CSW in AArch64) is executed which hits a dirty line in the L2 cache.
2) The interconnect accepts the dirty data, but does not return the write response.
3) An L2 linefill is started. This could be caused by an instruction fetch, a data read that does not allocate into L1, or an ACP read.
4) The linefill is to the same L2 index as the earlier cache clean, and picks the same line as a victim that the earlier cache clean accessed.
5) The interconnect returns the data for the L2 linefill.
6) The interconnect tries to return read data for other unrelated read requests, which the Cortex-A53 cluster will not accept by holding RREADYM low. This needs to be from a sufficient number of reads to fill all buffers on the path from the Cortex-A53 cluster through the interconnect or slave to the point where the read and write channels converge.
7) The interconnect or slave does not return the write response for the dirty line until the read data is accepted by the processor.

Implications

If the system contains dependencies as described above then when a sufficient number of outstanding transactions are present and certain timing conditions are met then the system could deadlock.

A system with ADB-400, CCI-400, and DMC-400 does not contain this dependency on the path to main memory. However, if a peripheral or different memory controller is connected to CCI-400 then those other components might introduce such a dependency. If this dependency is only present due to a peripheral in a non-cached memory region, then this erratum is unlikely to occur because:

- The L2 linefill with the incorrect dependency that causes this erratum must be to cacheable memory.
- Cache evictions can overtake non-cacheable and device writes within the cluster.
- The system must respond with the read data for the L2 linefill before the older writes are processed and responded to.
- The number of outstanding transactions to a peripheral is likely to be much less than to main memory.
- The number of outstanding reads the Cortex-A53 cluster can generate to device or non-cacheable memory is limited. If LDM instructions are not used to such peripherals then there will be a maximum of only a single read from each CPU in the cluster.

Workaround

For systems without a snoop filter in the interconnect, and without an L3 cache, this erratum can be worked around by disabling Evict and WriteEvict transactions. This can be done by clearing bit 14 and setting bit 3 in the L2ACTLR. This applies to CCI-400 based systems.

For systems with a snoop filter or L3 cache, it can be worked around by software that uses DCCMVAC, DCCMVAU, or DCCSW instructions replacing them with DCCIMVAC or DCCISW instructions, or their AArch64 equivalents.

827319: Data cache clean instructions might cause overlapping transactions to the interconnect

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2

Description

The Cortex-A53 MPCore processor supports connecting to an interconnect with a snoop filter or an L3 cache through an AMBA 5 CHI interface. It will notify the interconnect when a clean line is evicted from the cluster. Under certain conditions this erratum can cause a clean line eviction to occur at the same time as another transaction to the same address, which can cause data corruption if the interconnect reorders the two transactions.

Configurations affected

This erratum only affects configurations of the Cortex-A53 processor with an AMBA 5 CHI master interface and an L2 cache.

To be affected by this erratum, the system that Cortex-A53 is connected to must also contain a snoop filter and be capable of reordering requests to the same address.

Conditions

1) A DCCMVAC, DCCMVAU, or DCCSW instruction (or the equivalent DC CVAC, DC CVAU, or DC CSW in AArch64) is executed which hits a dirty line in the L2 cache.
2) The cluster sends a WriteClean transaction for the dirty data to the interconnect, but the interconnect does not yet return the response.
3) Another access to the L2 cache is made. This could be caused by an instruction fetch, an ACP access, or an eviction from the L1 data cache.
4) The access is to the same L2 index as the earlier cache clean, and picks the same line as a victim that the earlier cache clean accessed. This causes the cluster to send an Evict or WriteEvict transaction to the interconnect.
5) The interconnect reorders the transactions so that the Evict or WriteEvict is processed before the WriteClean.
6) Another master in the system accesses the same address, and the interconnect processes this request after it has processed the Evict or WriteEvict, but before it has processed the WriteClean.

Implications

If the interconnect reorders the two transactions to the same address then there will be a short period between the two transactions being processed where the snoop filter is incorrect. If another master request is processed during this time then it might not observe the dirty data.

CCN-504 contains a snoop filter and might reorder transactions. However it also contains an L3 cache, and if the L3 cache is enabled then there is an additional condition required to observe any failure. This condition is that the L3 line that is allocated by the WriteEvict transaction is then evicted from L3 before the WriteClean transaction is processed.

Workaround

This erratum can be worked around by software that uses DCCMVAC, DCCMVAU, or DCCSW instructions replacing them with DCCIMVAC or DCCISW instructions, or their AArch64 equivalents.

843819: Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The ARMv8 architecture requires that when all associated stages of translation are disabled for the current exception level, memory locations are only accessed due to instruction fetches within the same or next translation granule as an instruction which has been or will be fetched due to sequential execution. In the conditions detailed below, the Cortex-A53 MPCore processor might access other locations speculatively due to instruction fetches.

Configurations affected

All configurations of Cortex-A53 are affected.

Conditions

1) The processor must be executing at EL3, EL2 or Secure EL1.
2) The processor can be in either AArch32 or AArch64 execution state.
3) Address translation is disabled for the current exception level (by clearing the appropriate SCTLR.M, HSCTLR.M or SCTLR_ELx.M bit).
4) The HCR.VM/HCR_EL2.VM bit is set.

Implications

If these conditions are met, then speculative instruction fetches might be made to memory locations not permitted by the architecture.

Workaround

Because the HCR.VM bit is reset low, this situation is most likely to arise in powerdown code, if EL2 or EL3 software disables address translation before the core is powered down. To work around this erratum, software should ensure that HCR.VM is cleared before disabling address translation at EL3, EL2 or Secure EL1.

845719: A load might read incorrect data

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When executing in AArch32 state at EL0, a load to the same offset within a different 4GB region as a recent previous load in AArch64 state might read incorrect data.

Configurations affected

All configurations of Cortex-A53 are affected.

Conditions

The following sequence must occur for this erratum to be triggered:

Note the sequence requires execution in both AArch32 and AArch64, therefore at least one exception level change is required in this sequence.

1) At EL0 or EL1, in AArch32 or AArch64 state, a load, store, preload or data- or instruction-cache maintenance by MVA instruction is executed.
2) At EL0 or EL1 in AArch64 state, a load is executed with VA[63:32] != 32'h00000000.
3) At EL0 in AArch32 state, a load is executed to the same 4KB region as the instruction in position 1, with VA[31:6] matching the VA from the instruction in position 2.

Between position 1 and position 3 in the sequence, the following must **not** occur:

- A write to any of the following registers:
  - Any SCTLR.
  - Any TTBR.
  - Any TCR.
  - Any MAIR.
  - CONTEXTIDR.
- A TLB maintenance instruction is executed, followed by a DSB.

Between position 2 and position 3 in the sequence, the following must **not** occur:

- An address translation instruction is executed.

Implications

If the above conditions are met, then data corruption could occur. The load at EL0 could access data written at EL1 for which it does not have read permissions, however a load in non-secure state cannot access secure data.

Workaround

The erratum can be avoided by causing one of the conditions given in the list above to occur between positions 2 and 3 in the sequence. Because there must be an exception return from AArch64 to AArch32 between positions 2 and 3 in the above sequence, the recommended workaround is to insert a write of the CONTEXTIDR_EL1 register into operating system exception return sequences.

853175: APB access to ETM space while core is in Retention will never complete

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When a core is in Retention, it is in a low power state, and the main clock is turned off. An APB request to the ETM should cause the core to exit Retention and respond to the request.

Because of this erratum, the core will exit Retention, but **PREADYDBG** will not be driven HIGH, and the APB request will never complete.

Configurations Affected

All configurations are affected.

Conditions

1) The core must be in Retention.
2) A master must make an APB request to ETM space.

Implications

The core associated with the ETM will exit Retention normally, and will be able to execute instructions.

The APB transaction will not have any effect on the ETM.

The master that made the APB request will deadlock.

Another master in the cluster that makes an APB request to any slave will deadlock.

Workaround

The external debug or self-hosted debug software must disable Retention by setting CPUECTLR.CPURETCTL to 0x0 before accessing the ETM.

855871: ETM does not report IDLE state when disabled using OSLOCK

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The OS Lock feature in the ETM allows software running on a processor to disable external debug access and then save the register state before powering down the ETM. There is a defined sequence which must be followed to ensure that the register state is stable and that all trace has been output before the system is powered-down. Because of this erratum, when the OS Lock mechanism is used, the ETM will never indicate that it is safe to be powered-off.

Configurations Affected

All configurations are affected.

Conditions

1) The ETM is enabled using TRCPRGCTLR.EN == 1
2) The OS Lock feature is used to disable the ETM using TRCOSLAR.OSLK == 1

Implications

Software which follows the defined save and restore sequence will poll TRCSTATR.IDLE, but this will remain HIGH even though the ETM will be disabled and will drain. If the ETM was already disabled with TRCPRGCTLR.EN == 0 then the sequence will behave correctly.

Workaround

After the ETM has been disabled using TRCOSLAR.OSLK and the state of TRCPRGCTLR.EN has been recorded, TRCPRGCTLR.EN can be written LOW. This will allow the disable sequence to complete correctly.

855873: An eviction might overtake a cache clean operation

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The Cortex-A53 processor supports instructions for cache clean operations. To avoid data corruption, the processor must ensure correct ordering between evictions and cache clean operations for the same address.

Because of this erratum, the processor might issue an eviction and an L2 cache clean operation to the interconnect in the wrong order. The processor might also issue the transactions such that they are outstanding in the interconnect at the same time. This violates the ACE protocol specification and might cause the transactions to be erroneously re-ordered in the interconnect.

Configurations Affected

To be affected by this erratum, the following must be true:

- The processor is configured with an ACE bus interface.
- The processor is configured with an L2 cache.

Conditions

The erratum can be hit if the following conditions are met under specific timing conditions.

1) One or both of the following are true:
    1. L2ACTLR[14] is set to 1. This enables sending of WriteEvict transactions on the ACE interface when the processor evicts data that it holds in the UniqueClean state.
    2. L2ACTLR[3] is set to 0. This enables sending of Evict transactions on the ACE interface when the processor evicts clean data.
2) A core executes a cache clean by address operation for a line that is present and dirty in the L2 cache.
3) A core performs a memory access to the same set. This could be any type of memory access including a pagewalk, an instruction fetch, a cache maintenance operation, or a data access.
4) The instruction in condition (3) triggers an L2 cache eviction.
5) The line chosen for eviction from the L2 cache is the same line that was targeted by the cache clean operation in condition (2).

Implications

If the processor is connected to an interconnect that has a system cache or a snoop filter then this erratum might cause data corruption.

Workaround

The erratum can be avoided by upgrading cache clean by address operations to cache clean and invalidate operations. For Cortex-A53 r0p3 and later releases, this can be achieved by setting CPUACTLR.ENDCCASCI to 1. For earlier releases, software that uses `DCCMVAC` or `DCCMVAU` instructions can replace them with `DCCIMVAC` instructions.

859081: Some AT instructions executed from EL3 might incorrectly report a domain fault

Category B

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

Address translation instructions executed from EL3 and targeting EL1 or EL0 might report an incorrect result in the PAR when the HCR_EL2.DC bit is set.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

1) The core is executing at Exception level 3 in AArch64.
2) The core executes one of the following address translation instructions:
   - AT S1E0R, AT S1E0W
   - AT S1E1R, AT S1E1W
   - AT S12E0R, AT S12E0W
   - AT S12E1R, AT S12E1W
3) The Exception level targeted by the address translation instruction is Non-secure and AArch32.
4) HCR_EL2.DC is set to 1.
5) The DACR is programmed so that domain 0 would cause a domain fault if the HCR_EL2.DC bit had not been set. Note that this is the default value out of reset.

Implications

The PAR register will incorrectly report that a domain fault occurred.

Workaround

Secure software can set the DACR[1:0] to 0b01 before executing the address translation instruction. It should restore the previous DACR value before returning to a lower Exception level.

2.6. Category B (Rare)

855872: A Store-Exclusive instruction might pass when it should fail

Category B Rare

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The processor implements an internal exclusive monitor to manage Load-Exclusive, Store-Exclusive, and Clear-Exclusive instructions. Because of this erratum, a Load-Exclusive instruction to cacheable memory might set the monitor to the exclusive state when the processor does not have exclusive access to the line. A subsequent Store-Exclusive instruction might pass when it should fail.

The erratum affects all Load-Exclusive and Store-Exclusive instructions, including Load-Acquire Exclusive and Store-Release Exclusive instructions.

Configurations Affected

All configurations are affected.

Conditions

1) A core executes a store to memory that is marked as both inner-writeback and outer-writeback.
2) The store is not a Store-Exclusive (or a Store-Release Exclusive) or a Store-Release instruction.
3) The store is not followed by a DMB SY or DSB.
4) The store misses in the L1 data cache.
5) The store does not trigger a linefill. This requires one or more of the following to be true:
    - The core is in read-allocate mode.
    - The memory is marked as no-write-allocate.
    - The memory is marked as transient.
    - The store is a STNP instruction.
    - The store is triggered by a DC ZVA instruction.
6) The core starts a linefill to the same address as the store. The linefill is started for one of the following:
    - A PRFM, PLD, or PLDW instruction.
    - An automatic data prefetch.
    - A pagewalk.
7) The core executes a Load-Exclusive (or a Load-Acquire Exclusive) instruction to the same address as the store.
8) The store data is forwarded to the Load-Exclusive instruction.
9) The Load-Exclusive instruction retires before the linefill in condition (6) is serialised.

If the above conditions are met then the processor might set the internal exclusive monitor. This is not correct because the processor is not guaranteed to have exclusive access to the line.

Implications

If another core or master executes a Load-Exclusive instruction to the same address then both cores or masters might gain access to an exclusive region of code at the same time.

Most of the code sequences that can hit this erratum are not expected to exist commonly in real code. The scenario involving read-allocate mode in condition (5) is the most plausible.

The erratum has not been observed in the field. This indicates that the code sequences and timing conditions required to hit the erratum are rare.

Workaround

If a workaround is required then the only workaround is to avoid the conditions described. Disabling read-allocate mode by setting CPUACTLR.RADIS to 0b11 degrades write-stream performance. Therefore, the preferred workaround is to avoid conditions (2) or (3) using an appropriate Store-Release or DMB instruction.

859372: Advanced SIMD integer multiply instructions in IT block might cause data corruption

Category B Rare

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The Cortex-A53 processor can optionally be configured to support Advanced SIMD instructions.

The Cortex-A53 processor also supports conditional execution of instructions. In the T32 execution state, when the processor executes an IT instruction, up to four instructions at a time can become conditional.

Because of this erratum, if certain Advanced SIMD integer multiply instructions are executed conditionally, the processor might produce the wrong result for subsequent and dependent Advanced SIMD integer multiply-accumulate instructions.

Configurations affected

The processor must be configured with Advanced SIMD support.

Conditions

This erratum occurs when the processor executes a sequence of instructions in the T32 execution state. The sequence must comprise of the following instructions:

1) An `IT` instruction.
2) An optional instruction, belonging in the IT block.
3) An Advanced SIMD producer instruction, belonging in the IT block.
4) An optional instruction, belonging in the IT block.
5) An Advanced SIMD consumer instruction, which may or may not belong in the IT block.

Additionally, all the following conditions must hold:

- The producer instruction (3) must be one of the following:
    - VMLA (by scalar).
    - VMLA (integer).
    - VMLAL (by scalar).
    - VMLAL (integer).
    - VMLS (by scalar).
    - VMLS (integer).
    - VMLSL (by scalar).
    - VMLSL (integer).
    - VMUL (by scalar).
    - VMUL (integer).
    - VMULL (by scalar).
    - VMULL (integer).
- The consumer instruction (5) must be one of the following:
    - VMLA (by scalar).
    - VMLA (integer).
    - VMLAL (by scalar).
    - VMLAL (integer).
    - VMLS (by scalar).

  - VMLS (integer).
  - VMLSL (by scalar).
  - VMLSL (integer).
- The <Qd> or <Dd> destination operand of the producer instruction must match the <Qd> or <Dd> source operand of the consumer instruction.
- The <dt> data type of the producer instruction must be compatible with the <dt> data type of the consumer instruction.
- Both the producer instruction and the consumer instructions must be of the integer variant, and not of the floating-point or the polynomial variants.
- The producer instruction must fail its condition code check.
- The consumer instruction must either:
  - Not belong in the IT block and therefore execute unconditionally.
  - Belong in the IT block and pass its condition code check, because either:
    - The consumer has the opposite condition code compared to the producer.
    - The consumer has the same condition code compared to the producer, but a flag-setting instruction executes successfully in (4) and changes the condition flags.

Even though all the conditions above might hold, the erratum does not always occur. Specifically, any of the following conditions might affect its occurrence:

- The timing of the instructions in the processor pipeline.
- The arrangement of the instructions in dual-issuing pairs.

NOTE

Note that the ARM architecture deprecates the conditional execution of any instruction encoding provided by Advanced SIMD that is not also provided by floating-point. The architecture strongly recommends that such T32 Advanced SIMD instructions are never included in an IT block.

Implications

If this erratum occurs, the processor will incorrectly forward parts of the resulting value of the <Qd> or <Dd> operand from the producer instruction to the consumer instruction, even though the producer instruction has failed its condition code check. Therefore, the result of the consumer instruction will be wrong.

Workaround

A workaround is not expected to be required in most cases. This is because the erratum conditions require use of deprecated code.

If a workaround is required, the erratum can be avoided by disabling dual-issue. Dual-issue can be disabled by setting CPUACTLR.DIDIS to 0b1. Disabling dual-issue has a significant performance impact so this workaround is not recommended for most systems. If the erratum is encountered, it is preferable to work around it by avoiding the conditions described.

2.7. Category C

811819: A 64-bit load to Device memory which crosses a 64-byte boundary can access more data than is permitted

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0

Description

The Cortex-A53 MPCore processor implements the ARMv8-A architecture. One of the requirements of this architecture is that an implementation must not perform more accesses to a Device memory location than are specified by a simple sequential execution of the program.

On the Cortex-A53 processor, an LDRD or LDP of 32-bit general-purpose registers which is 32-bit aligned but crosses a 64-byte boundary will access more bytes than is architecturally permitted for the instruction.

Configurations Affected

All configurations.

Conditions

1) An LDRD or LDP of two 32-bit general-purpose registers is executed with an address 4 bytes before the end of a 64-byte region (thus addressing 32 bits at the top of one 64-byte region and 32 bits at the bottom of the next).
2) The address region is marked as Device memory.

If the above conditions are met then the processor will read the correct 32-bit chunk from the top of the first 64-byte region, but accesses 64 bits from the start of the next 64-byte region, rather than just the 32 bits the instruction explicitly accesses.

Implications

The effects of this erratum depend on the configuration of the system. It will only cause incorrect behavior when the system contains a read-sensitive peripheral accessed using Device memory, where the erratum causes a register access that the software did not intend. It is unlikely however that software would access such a peripheral using an unaligned instruction of the type described.

Workaround

Any software workaround depends on the specific peripheral being accessed, and how it is accessed. No workaround will be required if software does not use LDRD/LDP to access 64 bits of a peripheral starting at a 32-bit aligned address which crosses a 64-byte boundary.

814271: CPUMERRSR register might be updated incorrectly after multiple ECC/parity errors

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0

Description

The Cortex-A53 MPCore processor optionally provides support for soft error detection and correction on SRAM arrays, and for reporting of these events when they occur.

Because of this erratum, such errors might be reported incorrectly.

Configurations affected

Configurations which have the CPU_CACHE_PROTECTION configuration variable set to "TRUE" to provide protection on the CPU RAM arrays are affected.

Conditions

1) The CPUMERRSR is marked as not valid, either out of reset or because of a software write to the register.
2) A parity error is detected on the TLB RAM.
3) A parity or ECC error is detected on one of the DCU Tag RAMs, where the bottom two bits of the DCU RAM bank ID match the bank ID of the earlier error in the TLB RAM.

If these conditions are met, then the DCU Tag RAM error will cause the "Repeat error count" field in the CPUMERRSR to be incremented, rather than the "Other error count" field.

Implications

Software which uses the error count fields in the CPUMERRSR might see a higher number of TLB errors than is accurate, but will see a correct number of total errors.

Workaround

There is no workaround for this erratum.

822769: Mismatched aliases might cause deadlock

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1

Description

If the translation tables are constructed such that two or more different virtual addresses map to the same physical address, but the virtual addresses have different shareability attributes, then under certain conditions the processor might not respond to a snoop from the interconnect, which can lead to deadlock.

Configurations Affected

This erratum only affects configurations of the Cortex-A53 processor with an AMBA 4 ACE master interface, when it can receive snoops from a coherent interconnect.

Conditions

1) A load or store instruction, or ACP access, is executed to an address marked as cacheable shareable memory. This causes the line to be allocated into an L1 or L2 cache.
2) The interconnect supplies the cache line in a shared state, indicating that another master might also hold the cache line.
3) A full cache line write is made to the same address. This could be an ACP write of a full cache line, or one or more store instructions on a CPU that can be merged together into a full line.
4) If the full line write is from a CPU then it must not be allocated into the L1 cache, either because the memory is marked as transient or no-allocate in the page tables, the store is a non-temporal store, or the CPU has switched into read allocate mode after seeing several full cache line writes. The processor will send a MakeUnique coherency request externally.
5) The full line write is to cacheable non-shareable memory.
6) The full cache line causes an eviction of a different line from the L2 cache.
7) The interconnect makes a snoop request to the same address as the eviction.
8) The interconnect does not respond to the MakeUnique coherency request of the full cache line write until after the snoop completes.

If these conditions are met then the Cortex-A53 processor might not respond to the snoop until after the interconnect has completed and returned the response to the MakeUnique request, which could cause the system to deadlock.

Implications

A deadlock can occur during execution of software which explicitly sets up two or more aliases to the same physical memory page with different shareability attributes.

It is not expected that many applications make use of mismatched shareability attributes, and in most cases such use is architecturally UNPREDICTABLE.

Workaround

If multiple aliases to the same physical page in memory are required, the cache must be cleaned and invalidated for the relevant addresses between uses of the different aliases.

827971: Store release might not be multi-copy atomic

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3

Description

The Cortex-A53 MPCore processor supports store release instructions, which are required to be multi-copy atomic when observed by load acquire instructions. However, in some unusual code sequences, this erratum can cause the CPU executing the store release to observe the value stored earlier than it should have.

Configurations affected

All configurations are affected.

Conditions

On one CPU, the follow sequence must occur:

1) A store release instruction is executed to address A. This could be a STLR, STLRB, or STLRH instruction.
2) The store release is to normal cacheable memory, normal non-cacheable memory, or to Device-GRE memory.
3) If the store is to cacheable memory, it must miss in the cache and must not cause an L1 allocation. This means that either the memory is marked as no write allocate or as transient in the page tables, or the store release is immediately following a steam of stores which caused the processor to enter read allocate mode.
4) A store instruction is executed to address A. This must store the same number of or more bytes than the store release.
5) A load acquire instruction is executed to address A.
6) A store or store release to address B is executed.

On a second CPU, which can be within the same cluster or in a different cluster, the following sequence must occur:

1) A load or load acquire to address B is executed.
2) A load acquire to address A is executed.
3) An ordering dependency is introduced between the first load and the load acquire. This could be an address dependency, a barrier instruction, or the first load is a load acquire.

The ARM architecture requires that the store release executed to address A is not observed by load acquire instructions on the same CPU until it is also observable by any load acquire instructions on all other CPUs. However, if the above conditions are met and the timing conditions are such that the processor or interconnect reorders transactions to addresses A and B, then this erratum can cause the second processor to incorrectly observe the store to address B before it observes the store release to address A.

Implications

Any software that relies upon the ordering properties in the specific sequence above will have a small chance of observing accesses in an incorrect order, which might lead to a coherency failure.

However, it is not expected that typical synchronization code would rely on this, in particular it would not perform stores and store releases to the same address in close succession.

Workaround

Multi-processing code is not written in a style which exposes this erratum, and therefore is not expected to require a workaround. For any software that does require a workaround, the store to address A can be replaced by a store release.

831920: HCR.DC being set when HCR.VM not set does not correctly affect Data Cache Invalidate by MVA/VA behaviour

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3

Description

If a Cortex-A53 processor is configured with HCR.DC set and HCR.VM clear it will perform a Data Cache Invalidate by MVA/VA executed at non-secure EL1 as a Data Cache Clean and Invalidate, as required by the architecture, however it may cause the operation to be performed as an Invalidate on other masters in the same coherency domain or caches downstream of the Cortex-A53 processor.

Configurations Affected

This erratum affects all configurations of the Cortex-A53 processor.

Conditions

1) The HCR is written with the DC bit set and the VM bit clear.
2) The processor executes a DCIMVAC (AArch32) or DC IVAC (AArch64) at non-secure EL1, which targets an address which has stage 2 write permissions.
3) The address targeted is dirty in another master in the same coherency domain or a cache downstream of the Cortex-A53 processor.

If these conditions are met then the cache line targeted by the maintenance operation in step 2 will be invalidated in the caches described in step 3 without first having been cleaned.

Implications

A guest OS could cause data loss by invalidating a cached address without cleaning it.

Note that the combination of HCR.DC=1 and HCR.VM=0 was UNPREDICTABLE in the ARMv7 architecture, and as such existing hypervisor implementations are unlikely to encounter this erratum.

Workaround

Software can ensure that it always sets HCR.VM to 1 when setting HCR.DC to 1 (the architecture defines that HCR.VM behaves as if set to 1 when HCR.DC is set to 1, so this will have no other repercussions).

A hypervisor can always prevent a guest OS from invalidating an address by denying it stage 2 write permissions.

831921: AuxReg field of ID_MMFR0 reports incorrect value

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3

Description

The ARM architecture defines auxiliary fault status registers, ADFSR and AIFSR in AArch32. These are required to be implemented in the v8 architecture, and are implemented in the Cortex-A53 MPCore processor. However, due to this erratum, the AuxReg field in the ID_MMFR0 feature register indicates that these registers are not supported.

Configurations Affected

All configurations of Cortex-A53 are affected.

Conditions

There are no specific conditions for this erratum - any read of ID_MMFR0 in AArch32 or ID_MMFR0_EL1 in AArch64 will return the incorrect value of 0x1 for the AuxReg field, rather than 0x2 as would be expected.

Implications

It is not believed that this erratum will have any impact on system software. While the ADFSR and AIFSR are implemented on Cortex-A53, they always return zero on reads and ignore writes, so there will be no issues arising if software treats them as not implemented based on the ID_MMFR0.AuxReg value.

Workaround

There is no workaround for this erratum.

836870: Non-allocating reads might prevent a store exclusive from passing

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

If a Cortex-A53 processor is executing a load and store exclusive instruction in a loop then there are certain conditions that are allowed to cause the store exclusive instruction to repeatedly fail. This includes another processor repeatedly writing to the same cache line.

In addition to these allowed conditions, a non-allocating load from another processor might also cause the store exclusive instruction to repeatedly fail.

Configurations Affected

All configurations are affected.

Conditions

1) One CPU executes a loop containing a load exclusive and a store exclusive instruction. The loop will continue until the store exclusive instruction succeeds.
2) Another CPU, or master in the system, repeatedly executes a load to the same L1 data cache index as the store exclusive instruction.
3) The cache line containing the address of the load is present in the L1 cache of the CPU, and is not present in the L2 cache.
4) The load does not cause an allocation into the cache of the CPU or master executing the load. This must result in a snoop arriving at the CPU holding the cache line every time the load is performed.
5) The load is repeated at exactly the same frequency as the load and store exclusive loop, such that every time around the loop, the snoop arrives at the same time as the store exclusive instruction is executing.

A load can be non-allocating in Cortex-A53 in the following conditions:

1) A load instruction is executed on a CPU, while the CPUACTLR.DTAH bit is not set, with one of the following conditions:
    1. The memory address is marked as writeback cacheable, with no read allocate, in the pagetables.
    2. The memory address is marked as writeback cacheable, transient, in the pagetables.
    3. The memory address is marked as writeback cacheable, and an LDNP non-temporal load instruction is used.
2) A read transaction is made on the ACP interface.

Implications

If a CPU or other master is polling a location to determine when the value changes, then it can prevent another CPU from updating that location, causing a software livelock. However most polling routines would use memory that could be allocated into their cache, and for improved efficiency it is generally recommended to use a load exclusive and WFE instruction to avoid repeated polling. In addition, the frequency of the polling loop must match the frequency of the load store exclusive loop, so it is unlikely to get into this situation, and any other disturbance in the system such as an interrupt or other bus traffic could easily alter the frequency of either loop sufficiently to break the livelock.

Workaround

If the repeating load is being executed on a Cortex-A53 CPU then this erratum can be worked around by setting the CPUACTLR.DTAH bit. Note that from r0p4 and onwards, this bit is set by default.

If the repeating load is being executed by another master in the system connected to the ACP interface, then the erratum can be worked around by changing the frequency of the polling so that it no longer aligns with the frequency of the load and store exclusive loop.

If the repeating load is being executed by another master in the system connected through the coherent interconnect, then the erratum can be worked around either by ensuring that the other master allocates the line into its cache, or by changing the frequency of the polling so that it no longer aligns with the frequency of the load and store exclusive loop. Note that Cortex-A57 will always allocate the line into either the cache or a temporary buffer, and so will not require any workaround for typical polling loops.

836919: Write of JMCR in EL0 does not generate an UNDEFINED exception

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When EL0 is using AArch32 register width state and a write is performed in EL0 to JMCR, the write will be permitted but ignored. The intended behavior is that the instruction should be UNDEFINED.

Configurations Affected

All configurations are affected.

Conditions

1) The processor is executing in AArch32 User mode.
2) A write to the JMCR is executed, using the instruction `MCR p14,7,<Rt>,c2,c0,0`.

Implications

Rather than treating the `MCR` instruction as UNDEFINED, the write to the JMCR is ignored.

Workaround

There is no workaround for this erratum.

845819: Instruction sequences containing AES instructions might produce incorrect results

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

Certain sequences of instructions that include AES instructions might cause incorrect results to be generated when executed in AArch64 state on the Cortex-A53 processor.

Configurations affected

Only configurations of Cortex-A53 that include the Cortex-A53 Cryptography Extension are affected.

Conditions

There are two code sequences which can stimulate this erratum, both in AArch64 execution state.

Condition 1:

1) An AESE instruction is executed.
2) A USQADD instruction is executed.
   - Both Vn and Vd of this instruction must be the same register as Vd for the AES instruction.
   - The "size" field for this instruction must be '00', indicating byte-sized elements.
   - This can be either the "Vector" or "Scalar" form of the USQADD instruction.

Condition 2:

1) A SUQADD instruction is executed.
   - The "size" field for this instruction must be '00', indicating byte-sized elements.
   - This can be either the "Vector" or "Scalar" form of the SUQADD instruction.
2) An AESMC or AESIMC instruction is executed.
   - Both Vn and Vd of this instruction must be the same register as Vd for the SUQADD instruction.

For both these conditions, the two instructions must be executed consecutively, with no other instructions or exceptions between them. If either of these sets of conditions is met, the final result of the affected instruction sequence might be incorrect.

Implications

The sequences of instructions described in the conditions above are not expected to occur in real code, because they do not perform useful computation. As such, there is no impact expected to real systems.

Workaround

Because the code sequences for this erratum are not expected to occur in real code, no workaround is required.

851672: ETM might trace an incorrect exception address

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The address in an exception packet should be the preferred exception return address. Because of this erratum, the address might be equal to the target address of the exception. The trace stream is not corrupted, and decompression can continue after the affected packet.

Configurations Affected

All configurations are affected.

Conditions

The following sequence is required to hit this erratum:

1) The ETM must start tracing instructions because of one of the following:
   - Viewinst goes high.
   - The security state changes such that trace is now permitted.
   - The values of the external debug interface (**DBGEN**, **SPIDEN**, **NIDEN**, **SPNIDEN**) change such that trace is now permitted.
   - The core exits debug mode.
2) Before the core executes any other behavior which would cause trace to be generated, it executes a direct branch instruction, which might be taken or not-taken.
3) The next instruction is a load or store that takes a Data Abort or Watchpoint exception.

After this sequence, provided certain timing specific conditions are met, the address in the exception packet might be incorrect.

Implications

The trace decompressor might incorrectly infer execution of many instructions from the branch target to the provided address.

Workaround

The trace decompressor can detect that this erratum has occurred by checking if the exception address is in the Vector Table and the branch was not expected to be taken to the Vector Table.

A decompressor can infer the correct address of the exception packet. It will be given by the target of the preceding branch (If the branch was taken), or the next instruction after the branch (If the branch was not-taken).

851871: ETM might lose counter events while entering wfx mode

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

If the ETM resources become inactive because of low-power state, there is a one-cycle window during which the counters and the sequencer might ignore counter-at-zero resources.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

The following sequence is required to hit this erratum:

1) The core executes a WFI or WFE instruction.
2) The ETM enters low-power state because of this.
3) In a one-cycle window around this point, either:
    - A counter in self-reload mode generates a counter-at-zero resource.
    - A counter in normal mode gets a RLDEVENT on the cycle in which it has just transitioned to zero.
4) A counter or sequencer is sensitive to the counter-at-zero resource.

Implications

Counters sensitive to a counter-at-zero resource might not reload or decrement. If the sequencer is sensitive to a counter-at-zero resource, it might not change state, or might change to an incorrect state.

Workaround

The ETM can be prevented from entering low-power mode by programming LPOVERRIDE bit of TRCEVENTCTL1R to 1. This workaround is only needed if there is a counter or sequencer sensitive to a counter-at-zero resource, and is not normally necessary.

852071: Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

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-A53 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-A53 does not force completion of the packet prior to acknowledging the flush request. This is a violation of the ETMv4 architecture.

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.

852521: A64 unconditional branch might jump to incorrect address

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When executing in AArch64 state with address translation disabled, unconditional immediate branch instructions might jump to an incorrect address.

Configurations Affected

All configurations of Cortex-A53 are affected.

Conditions

1) The processor is executing in AArch64 state.
2) The SCTLR_ELx.M bit for the current exception level is 0.
3) The HCR_EL2.VM bit is 0.
4) A B or BL instruction from the "Unconditional branch (immediate)" encoding class is executed.
5) This branch has an imm26 field of 0x1FFFFFF, encoding a branch target of {pc}+0x7FFFFFC.

Implications

If these conditions are met, then the processor might incorrectly branch to the target {pc}-0x8000004 instead of {pc}+0x7FFFFFC.

Workaround

The workaround for this erratum is to avoid the conditions described.

853172: ETM might assert AFREADY before all data has been output

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When the AFVALID signal on the ATB interface is asserted, the ETM should immediately start outputting all buffered trace. It should assert the AFREADY output one cycle after all trace that was buffered on the cycle in which AFVALID was first asserted.

Because of this erratum, the AFREADY signal might be asserted before all the necessary trace has been output.

Configurations Affected

All configurations are affected.

Conditions

The ETM must contain buffered trace.

Implications

This might result in the ETM containing trace that was generated before the flush request when the rest of the system expects this trace to have been output.

Workaround

The system can ensure that all trace has been drained from the ETM by disabling it by setting TRCPRGCTLR.EN to 0. The system should then poll the TRCSTATR.IDLE bit, and once it reads as 1, then the ETM is idle, and all trace that was generated before the write to TRCPRGCTLR has been output.

855827: PMU counter values might be inaccurate when monitoring certain events

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The Cortex-A53 processor implements a Performance Monitor Unit (PMU). The PMU allows programmers to gather statistics on the operation of the processor during runtime. Because of this erratum, the PMU counter values might be inaccurate when monitoring certain events. Specifically:

- The INST_RETIRED event should count architecturally executed instructions. Because of this erratum, it might count more instructions than were architecturally executed.
- The ST_RETIRED event should not count Store-Exclusive instructions that fail. Because of this erratum, it does count these instructions.
- The UNALIGNED_LDST_RETIRED event should count loads and stores that fail their alignment check. Because of this erratum, it might also count LDRD and STRD instructions that pass their alignment check.
- The EXC_TAKEN and EXC_RETURN events should be filtered precisely according to the Exception level/security state they were executed in. Because of this erratum, they will be filtered according to the destination Exception level/security state.
- The BUS_ACCESS_ST and BUS_ACCESS events should count bus accesses per core. Because of this erratum, they might be counted incorrectly.

Configurations Affected

All configurations are affected.

Conditions

1) A performance counter is enabled and configured to count one of the following events:
    - INST_RETIRED.
    - ST_RETIRED.
    - UNALIGNED_LDST_RETIRED.
    - EXC_TAKEN.
    - EXC_RETURN.
    - BUS_ACCESS_ST.
    - BUS_ACCESS.
2) This condition depends on which event the performance counter is configured to monitor:
    - INST_RETIRED: An immediate branch instruction is executed.
    - ST_RETIRED: A Store-Exclusive instruction is executed, and fails.
    - UNALIGNED_LDST_RETIRED: A word-aligned but not doubleword-aligned LDRD or STRD instruction is executed.
    - EXC_TAKEN: An exception is taken.
    - EXC_RETURN: An exception return is executed.
    - BUS_ACCESS_ST or BUS_ACCESS: A write or eviction occurs on the bus.
3) For INST_RETIRED, ST_RETIRED, and UNALIGNED_LDST_RETIRED, the filtering settings for the performance counter are set so that an event should be counted if it occurs. For EXC_TAKEN and EXC_RETURN, the filtering settings are set so that:
    - Events should be counted in the original Exception level/security state, but should not be counted in the Exception level/security state following the exception or the exception return, or
    - Events should not be counted in the original Exception level/security state, but should be counted in the Exception level/security state following the exception or the exception return.

Implications

If the erratum conditions are met, the performance counter might increment when it should not or it might not increment when it should. Specifically:

- INST_RETIRED: The counter might erroneously increment by two when only one instruction is executed. This can occur when an immediate branch is executed in combination with other microarchitectural conditions. The conditions are sufficiently rare that the final counter value is not expected to be impacted significantly in most scenarios.
- ST_RETIRED: The counter will erroneously increment for the failed Store-Exclusive instruction. Store-Exclusive instructions are not expected to form a significant part of the total number of stores.
- UNALIGNED_LDST_RETIRED: The counter will erroneously increment for the LDRD or STRD instruction. LDRD and STRD are commonly used when moving large blocks of memory. However, they are usually doubleword aligned.
- EXC_TAKEN and EXC_RETURN: The counter will erroneously increment or erroneously fail to increment. This is typically a less interesting event for performance measurement. It is low impact because filtering by destination gives an accurate result. Filtering is controlled using the upper bits of the **PMEVTYPER** registers.
- BUS_ACCESS_ST and BUS_ACCESS: The counter will erroneously increment or erroneously fail to increment. The inaccuracy varies between cores. Some bus accesses for core 0 might be attributed to core 1. Some bus accesses for core 1 might be attributed to core 2 or core 3. Bus accesses for cores 2 and 3 will not be attributed to any core. The number of cores present in the configuration does not affect how the bus accesses are attributed. For example, some core 1 accesses might be attributed to cores 2 and 3 even if the configuration has only two cores. The impact on the final counter value in each core is high.

This might lead to inaccurate results when using the PMU to debug or profile code.

Workaround

For the EXC_TAKEN and EXC_RETURN events, the erratum can be worked around by changing the filtering settings for the performance counter to monitor the destination Exception level/security state instead of the Exception level/security state that the exception or exception return are executed in.

The BUS_ACCESS and BUS_ACCESS_ST events can be counted accurately for core 0 by enabling the counter on both core 0 and core 1 and taking the total. This workaround requires core 1 to be present in the configuration and idle during testing. Similarly, the total count for core 0 and core 1 can be found by enabling the counter on all four cores and taking the total. This workaround requires cores 2 and 3 to be present in the configuration and idle during testing. In a configuration with four cores, the events can be used with an intensive memory test that runs on two cores to give a reasonable indication of maximum bandwidth for the cluster.

The event L2D_CACHE_WB counts write-backs from the L2 cache that are attributable to a core. For a test focused on cacheable memory bandwidth, this might be a suitable replacement for BUS_ACCESS_ST. This event includes:

- Write-backs as a result of evictions and cache maintenance instructions.
- Writes in read allocate mode (write-streaming mode).

L2D_CACHE_WB does not include:

- Write-backs as a result of snoop requests from outside of the cluster.
- Write-backs as a result of ACP interface accesses.

There is no workaround for the INST_RETIRED, ST_RETIRED, and UNALIGNED_LDST_RETIRED PMU events.

855829: Reads of PMEVCNTR<n> are not masked by HDCR.HPMN

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The HDCR.HPMN/MDCR_EL2.HPMN field can be set by software executing at EL2 to restrict access to the performance counters by Non-secure EL0/EL1 software to a subset of the counters. If n > HPMN, then performance counter n is not accessible to Non-secure EL0/EL1. Because of this erratum, on Cortex-A53 reads of inaccessible registers might not return zero as expected, but instead might read the actual contents of the counter registers.

Configurations Affected

All configurations of Cortex-A53 are affected.

Conditions

1) The processor is executing at Non-secure EL1 or EL0.
2) HDCR.HPMN/MDCR_EL2.HPMN is set to a value n, where n < 6.
3) A read of PMEVCNTR<n>/PMEVCNTR<n>_EL0 is executed.

Implications

If the conditions described above are met, then Non-secure EL0/EL1 software can read the values of performance counters reserved for EL2.

Workaround

Software executing at EL2 can set the HDCR.TPM/MDCR_EL2.TPM bit. This will cause all Non-secure EL0 and EL1 accesses to performance monitor registers to trap to EL2. The EL2 software can then emulate accesses as required, and can mask out accesses to reserved registers.

855830: Loads of mismatched size might not be single-copy atomic

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The Cortex-A53 processor supports single-copy atomic load and store accesses as described in the ARM architecture. However, in some unusual code sequences, this erratum can cause the CPU executing a store and later a load to the same address but with a different access size to load data that does not meet the requirements of a single-copy atomic load.

Configurations affected

All Cortex-A53 configurations are affected.

Conditions

On one CPU, the following sequence must occur:

1) A store instruction is executed. This could be any halfword, word, or doubleword store instruction that is not a store release, and the address must be aligned to the access size.

On a second CPU, which can be within the same cluster or in a different cluster, the following sequence must occur:

1) A store instruction is executed. This store must be a smaller access size to the store on the first CPU, and must be to an address with the bytes accessed by the first CPU. The address must also be aligned to the access size.
2) The store instruction must not allocate into the cache. This could be because:
   - The memory address is marked as transient, or
   - The write allocate hint in the translation table is not set, or The memory is marked as non-cacheable, or
   - The CPU has recently executed a stream of stores and so has dynamically switched into a no write allocate mode.
3) A load instruction is executed. The load must be a larger access size than the store from the same CPU, and at least some bytes of the load must be to the same address as the store. The address must also be aligned to the access size.

The ARM architecture requires that the load is single-copy atomic. However, in the conditions described, the load might observe a combination of the two stores, indicating that the store on the first CPU was serialized first. However, if the load is repeated, then the second time it might see just the data from the store from the first CPU indicating that the store on the first CPU was serialized second.

Implications

Concurrent, unordered stores are not common in multi-threaded code. In the C11 standard, they are restricted to the family of "relaxed" atomics. In addition, using different size load and store instructions to access the same data is unusual. Therefore the majority of multi-threaded software is not going to meet the conditions for this erratum.

Workaround

Most multi-threaded software is not expected to meet the conditions for this erratum and therefore will not require a workaround. If a workaround is required, then the store on the second CPU should be replaced with a store release instruction.

859072: ETM might output an incorrect exception return address

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The processor supports instruction trace using a per-core ETM. If tracing is enabled, a core's ETM generates an Exception element when the core takes an exception. The Exception element contains an indication that an exception has occurred, the type of exception, and the exception return address. Because of this erratum, the exception return address that is output by the ETM might have bits [63:49] set to 1 when they should be set to 0.

Configurations Affected

All configurations.

Conditions

1) Trace is enabled.
2) Non-invasive debug is permitted.
3) A core is in the AArch64 execution state.
4) The core executes one of the following instructions at VA 0x0000_ffff_ffff_fffc
   - SVC
   - HVC
   - SMC

Implications

The exception return address reported by the ETM might be incorrect. For example, an SMC instruction at VA 0x0000_ffff_ffff_fffc might cause the ETM to output an exception return address of 0xffff_0000_0000_0000 instead of 0x0001_0000_0000_0000.

Real code is not expected to meet the conditions in this erratum because, if this erratum did not exist, the core would access a VA that is outside of the implemented virtual address space.

Workaround

There is no workaround for this erratum.

859074: WFx can cause the PC to jump from lower VA subrange to upper VA subrange

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

If a core executes a WFI or WFE instruction at VA 0x0000_ffff_ffff_fffc in the AArch64 execution state then, when the core wakes up, it should resume execution from VA 0x0001_0000_0000_0000. This address is outside of the implemented virtual address space and will trigger a Translation fault if accessed.

Because of this erratum, the core will resume execution from VA 0xffff_0000_0000_0000 instead of 0x0001_0000_0000_0000. Address 0xffff_0000_0000_0000 is a valid address in the upper VA subrange for EL0/EL1 translations if TCR_EL1.T1SZ is set to 16.

Configurations Affected

All configurations.

Conditions

1) A core is in the AArch64 execution state.
2) The core executes a WFI or WFE instruction at VA 0x0000_ffff_ffff_fffc.

Implications

Real code is not expected to include a WFI or WFE instruction at VA 0x0000_ffff_ffff_fffc because, if this erratum did not exist, wake-up from the WFx would always trigger an access to a VA that is outside of the implemented virtual address space.

Workaround

If a workaround is required, code could be inserted at VA 0xffff_0000_0000_0000 to catch unexpected accesses to this address.

859075: Accessing EDPCSR has side-effects when OS Lock is locked

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

Access to the External Debug Program Counter Sample Register, EDPCSR, should not have side-effects when the OS lock is locked or when the OS Double Lock is locked. Because of this erratum, the side-effects are only disabled when OS Double Lock is locked.

Configurations Affected

All configurations are affected.

Conditions

1) The OS Lock is locked (EDPRSR.OSLK == 1).
2) The OS Double Lock is unlocked (EDPRSR.DLK == 0).
3) An external debugger reads EDPCSR[31:0].

Implications

If the erratum conditions are met then EDCIDSR, EDVIDSR, and EDPCSR[63:32] will be updated as if the OS Lock were unlocked.

Workaround

There is no workaround.

859076: Instruction cache parity error might cause an incorrect abort to be taken

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The core cache protection option allows the processor to detect and correct a 1-bit error in the instruction cache RAMs. If an error is detected on an instruction at the end of a page, and the following page would generate a fault if executed, then under some conditions it is possible for a fault to be taken when no instruction from the faulting page was architecturally executed.

Configurations affected

This erratum affects configurations of the processor with CPU_CACHE_PROTECTION enabled.

Conditions

1) The core is executing in AArch32 T32 instruction state.
2) The program counter points to a 16-bit length instruction which is at the end of a page.
3) The related instruction is stored in the L1 instruction cache.
4) There is a single bit error on one specific bit of the instruction, which causes the instruction to be incorrectly interpreted as a 32-bit instruction.
5) The page which is immediately after the address range of the previous 16-bit instruction would cause a translation fault, access flag fault, or permission fault if it was accessed.

Implications

If the above conditions are met, then the translation, permission, or access flag fault is incorrectly taken, instead of correcting the error. There is still substantial benefit being gained from the parity logic. There might be a negligible increase in overall system failure rate due to this erratum.

Workaround

No workaround is required.

859077: Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When reporting a synchronous uncorrectable ECC error, an incorrect encoding is written to the fault status code field of the Instruction Fault Status Register.

Configurations Affected

This erratum affects configurations of the processor that have CPU_CACHE_PROTECTION or SCU_CACHE_PROTECTION set to TRUE.

Conditions

1) A core is in the AArch32 execution state.
2) The core is executing in EL0, EL1, or EL3.
3) The core is using the Short-descriptor page format.
4) An uncorrectable ECC error occurs in one of the following caches:
   - The L1 data cache of the core.
   - The L1 data cache of another core in the processor.
   - The L2 cache.
5) An instruction fetch or a translation table walk for an instruction fetch hits the cache line containing the ECC error, triggering a prefetch abort.

If these conditions are met, then IFSR.FS[4] is set to 0 instead of 1.

Implications

When the short descriptor page format is in use, software cannot make a distinction between a Domain fault, level 1, or a synchronous parity or ECC error on memory access. Both will be reported in the IFSR as 01001.

Similarly, software cannot make a distinction between synchronous external abort on translation table walk and synchronous parity or ECC error on translation table walk. Both will be reported in the IFSR as synchronous external abort on translation table walk (IFSR.FS[4:0] will be 01100 for translation level 1 and 01110 for translation level 2). This is valid behavior because it is IMPLEMENTATION DEFINED whether ECC errors are reported using the assigned fault code or using another appropriate encoding. However, the behavior is inconsistent with fault code usage in other fault status registers, and in the IFSR when using the Long-descriptor page format.

Workaround

When reading the fault status code from the IFSR, software will have to be aware that the value reported might not be accurate. If software reads 01001 for IFSR.FS[4:0], then software will have to determine if a Domain fault could have occurred for this address.

859080: ETM trace reports incorrect branch target

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

When executing a branch that targets an out of range virtual address, the ETM might trace the target address incorrectly.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

1) ETM trace is enabled and not prohibited.
2) The core is executing code located in the virtual address range 0hFFFF000000000000-0hFFFFFFFFFFFFFFFF, and bit[48] of the PC is set. If the relevant TCR.TBI bit is set, then bits[63:56] of the virtual address can be any value.
3) The core executes a branch instruction with the target virtual address in the range 0h0002000000000000-0hFFFEFFFFFFFFFFFF, with bit[48] of the target address clear. This could be any kind of instruction that alters the program flow, including immediate branches, register based branches, and returns.

Implications

The target of the branch will take a translation fault, because it is not in a valid address range, but because the ETM trace decompression will report the branch target incorrectly the reason for the fault might be harder to determine when debugging. Bits[47:0] of the branch target will still be correct, but bits[63:48] will be incorrectly reported as all zero.

Workaround

There is no workaround for this erratum.

859082: ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

An ATS12NSOPR address translation instruction executed from EL3 might report an incorrect result in the PAR when both the HCR.TGE bit is set and the SCTLR.M is set.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

1) The core is executing at Exception level 3 in AArch32.
2) SCR.NS = 0
3) HCR.TGE = 1
4) SCTLR(ns).M = 1
5) The core executes an ATS12NSOPR instruction.

Implications

The PAR register will incorrectly report the translation as if the stage 1 MMU was enabled.

Note that the combination of both HCR.TGE and SCTLR.M bits being set was UNPREDICTABLE in earlier versions of the architecture, and was only given a defined behavior to reduce the UNPREDICTABLE space. It is not expected to be a useful combination for software.

Workaround

Secure software can clear the SCTLR(ns).M bit before executing the address translation instruction, if the HCR.TGE bit is set. It should restore the previous SCTLR(ns).M value before returning to a lower Exception level.

859083: Debug not entering Memory Access mode without setting EDSCR.ERR

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The processor supports entering debug Memory Access (MA) mode through the APB interface. This might fail when an instruction previously executed through the APB interface has not yet completed.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

1) The debugger inserts an instruction through the APB interface by writing the EDITR.
2) The debugger writes to the EDSCR to enter MA mode.
3) The debugger reads the DTRTX at the same time as the instruction completes.

Implications

The APB read of the DTRTX does not start up the MA state machine, but EDSCR.TXU and EDSCR.ERR are not set.

Workaround

Before executing the instruction, the debugger should write 1 to the EDRCR.CSPA. After executing the instruction, the debugger should poll the EDSCR.PipeAdv bit to determine when the instruction has completed, and only attempt to enter MA mode after the PipeAdv bit is set.

859373: Mismatch between EDPRSR.SR and EDPRSR.R

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The processor provides access to the EDPRSR through the APB interface. If this access is done at the same time as the core leaves a Warm reset, then a subsequent read of the same register will read an incorrect value of the SR field.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

1) The core is in Warm reset.
2) A debugger reads the EDPRSR register over the APB interface.
3) The core comes out of Warm reset during the APB read.
4) A second APB read is made to the EDPRSR register.

Implications

The first read of the EDPRSR will read the SR field and R field both as 0b1.

The second read of the EDPRSR.SR field will read 0b0 whereas the previous read of the EDPRSR.SR was 0b1 while in Warm reset. Because the first read took place while in Warm reset, the sticky bit should still be set on the second read.

Workaround

If the debugger reads the EDPRSR and sees both the SR and R fields set, then it must remember this result and on the next EDPRSR read treat the SR bit as if it was set.

859374: AArch64 memory streaming performance might be lower than expected when using a 64KB translation granule

Category C

Products Affected: Cortex-A53 MPCore.

Present in: r0p0, r0p1, r0p2, r0p3, r0p4

Description

The processor includes an automatic data prefetcher to improve memory streaming performance. Because of this erratum, the data prefetcher does not operate as intended when virtual addresses are translated using a 64KB translation granule.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

1) A core is in the AArch64 execution state.
2) TCR_ELx.{TG0 or TG1} is set to 01 for the current Exception level and TTBR combination, indicating a 64KB translation granule.
3) One of the following is true:
    1. The current Exception level is Secure EL1, EL2, or EL3.
    2. The current Exception level is Non-secure EL1 or EL0 and Stage 2 translation is not enabled.
    3. The current Exception level is Non-secure EL1 or EL0 and VTCR_EL2.TG0 is set to 01, indicating a 64KB translation granule.
4) The core executes a sequence of loads or stores that triggers the automatic data prefetcher. For example, a sequence of loads or stores to consecutive cachelines.

Implications

The automatic data prefetcher will not prefetch data effectively.

This will result in low AArch64 memory streaming performance for some workloads but it will not cause functional failures.

Workaround

AArch64 memory streaming performance can be improved by using the 4KB translation granule.