ARM Processor

Cortex®-A57 MPCore ®

Product Revision r0, r1

Software Developers Errata Notice

Non-Confidential - Released

ARM-EPM-049219 v23 Released

Non-Confidential Proprietary Notice

Web Address

http://www.arm.com

Feedback on content

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

- the document title
- the document number, ARM-EPM-049219
- 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.

24 Feb 2020: Changes in Document v23

Page	Status	ID	Cat	Rare	Summary of Erratum
35	New	1742098	CatB		ELR recorded incorrectly on interrup taken between cryptographic instructions in a sequence
88	New	1742089	CatC		Writes to normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode
87	New	1742085	CatC		Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort
85	Updated	1572968	CatC		Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort

20 Sep 2019: Changes in Document v22

Page	Status	ID	Cat	Rare	Summary of Erratum
33	New	1319537	CatB		Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation
34	New	1387217	CatB		DSB is insufficient to ensure page table entries being validated are visible to subsequent translations
80	New	681545	CatC		Trace On packets from ETM are not generated in specific conditions around System Error exceptions
81	New	1185470	CatC		Exception packet for return stack match might return incorrect [E1:E0] field
82	New	1328360	CatC		Speculative TLB fills might occur past a DSB Instruction
83	New	1406397	CatC		TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0
84	New	1572968	CatC		Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort

31 Aug 2017: Changes in Document v21

Page	Status	ID	Cat	Rare	Summary of Erratum
40	New	859972	CatB	Rare	Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue

15 Jan 2016: Changes in Document v20

Page	Status	ID	Cat	Rare	Summary of Erratum
40	New	836019	CatB	Rare	Cortex-A57 might deadlock when performing multiple non-overlapping stores with certain memory types and specific timing
31	New	853871	CatB		Distributed Virtual Memory operations during hardware flush might cause deadlock
80	New	853971	CatC		An external data snoop might cause data corruption when an Evict transaction is pending

09 Oct 2015: Changes in Document v19

Page	Status	ID	Cat	Rare	Summary of Erratum
30	New	852523	CatB		Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings
76	New	851023	CatC		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores
77	New	852579	CatC		Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions
78	New	852580	CatC		Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement
79	New	852581	CatC		Trace Context packet might not be output on a Reset or System Error exception

16 Jun 2015: Changes in Document v18

Page	Status	ID	Cat	Rare	Summary of Erratum
74	New	850774	CatC		Cortex-A57 incorrectly allows access to GICv3 common registers in a specific configuration
75	New	850775	CatC		ATB stall from trace subsystem might deadlock the processor

27 Feb 2015: Changes in Document v17

Page	Status	ID	Cat	Rare	Summary of Erratum
73	New	844270	CatC		Syndrome value incorrect for software-induced Virtual Abort exception

18 Sep 2014: Changes in Document v16

Page	Status	ID	Cat	Rare	Summary of Erratum
28	New	833471	CatB		VMSR FPSCR functional failure or deadlock
29	New	834921	CatB		Code sequence continuously hitting the L1 cache can block snoop
39	New	834220	CatB	Rare	Stage 2 translation fault might be incorrectly reported in presence of a stage fault
68	New	833472	CatC		PMU events 0x73 and 0x77 are not incremented for certain instructions
69	New	834221	CatC		Trace packet might be corrupted in certain cases
70	New	834469	CatC		PMU event counters are incorrectly permitted to count events
71	New	834528	CatC		PMU event BUS_CYCLES might be incorrect in some cases
72	New	834920	CatC		Write of JMCR in EL0 does not generate an undefined exception

08 Aug 2014: Changes in Document v15

Page	Status	ID	Cat	Rare	Summary of Erratum
27	New	832075	CatB		Cortex-A57 might deadlock when WB exclusive load/store instructions are mixed with device loads
64	New	832076	CatC		AuxReg field of ID_MMFR0 reports incorrect value
65	New	832569	CatC		Cortex-A57 L1 prefetch works inefficiently for a load stream whose address is mapped to a page size of 64K or larger
66	New	832769	CatC		HSTR.{T7,T8,T15} and HSTR_EL2.{T7,T8,T15} bits incorrectly trap CDP instructions
67	New	833069	CatC		Disabling MMU Translation with CPUACTLR_EL1 "Enable Invalidates of BTB" bit set can cause Invalidate by PA or VA to fail

10 Jun 2014: Changes in Document v14

Page	Status	ID	Cat	Rare	Summary of Erratum
25	New	829520	CatB		Code bounded by indirect conditional branch might corrupt instruction stream

16 May 2014: Changes in Document v13

Page	Status	ID	Cat	Rare	Summary of Erratum
38	New	828024	CatB	Rare	TLB maintenance operations might not be synchronized by DSB instruction
63	New	828023	CatC		Cortex-A57 might violate read-after-read memory ordering when the memory accesses of loads are not single-copy atomic

28 Apr 2014: Changes in Document v12

Page	Status	ID	Cat	Rare	Summary of Erratum
24	New	826977	CatB		L2 Data RAM corruption might occur because of a streaming write to Device-GRE memory which detects a hazard
37	New	826974	CatB	Rare	Cortex-A57 might violate memory order between an older store and a younger load when executing DMB ALL/ALL instruction
60	New	825570	CatC		Access to Debug EDSCR.STATUS[5:2] register bits is Read-Write instead of Read-Only
61	New	826975	CatC		Cortex-A57 might livelock because of exclusive load to non-shared memory location with mismatched attributes
62	New	826978	CatC		Cortex-A57 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary

13 Mar 2014: Changes in Document v11

Page	Status	ID	Cat	Rare	Summary of Erratum
56	New	822224	CatC		A57 might deadlock or have data corruption if the page attribute of a current page is remapped by another processor
57	New	822227	CatC		Using unsupported 16K translation granules might cause Cortex-A57 to incorrectly trigger a domain fault
59	New	823969	CatC		Halting debug event occurring while ITSTATE is non-zero might cause subsequent instructions from the ITR to execute

05 Feb 2014: Changes in Document v10

Page	Status	ID	Cat	Rare	Summary of Erratum
58	New	822369	CatC		HSR.EC might be incorrect for certain accesses when HCR.TGE=1

28 Jan 2014: Changes in Document v9

Page	Status	ID	Cat	Rare	Summary of Erratum
54	New	821570	CatC		Read following a write of a Timer TVAL register might return incorrect value
55	New	821571	CatC		Read or write of ICC_SRE_EL1(NS) might incorrectly generate an undefined exception

19 Dec 2013: Changes in Document v8

Page	Status	ID	Cat	Rare	Summary of Erratum
22	Updated	817169	CatB		DSB instruction in processor power down sequence might not drain all required transactions
46	Updated	811672	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned

26 Nov 2013: Changes in Document v7

Page	Status	ID	Cat	Rare	Summary of Erratum
22	New	817169	CatB		DSB instruction in processor power down sequence might not drain all required transactions
23	New	817171	CatB		AMBA 5 CHI systems using SCLKEN may hang following link deactivation
51	New	817170	CatC		Continuous streaming stores to device memory might block older stores
52	New	817172	CatC		Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption
53	New	817722	CatC		Read after read memory ordering requirements might be violated with specific sequences of exclusive stores and loads

01 Nov 2013: Changes in Document v6

Page	Status	ID	Cat	Rare	Summary of Erratum
18	Updated	811671	CatB		A57 issues multiple concurrent DVMOp(Sync) transactions on AMBA 5 CHI interconnect
50	Updated	814670	CatC		A57 DMB barrier does not guarantee observation of the effects of a cache maintenance operation

15 Oct 2013: Changes in Document v5

Page	Status	ID	Cat	Rare	Summary of Erratum
49	New	814669	CatC		L2 ECC error might cause a device memory type load to stall until the next timer interrupt
50	New	814670	CatC		A57 DMB barrier does not guarantee observation of the effects of a cache maintenance operation

02 Oct 2013: Changes in Document v4

Page	Status	ID	Cat	Rare	Summary of Erratum
20	New	813419	CatB		TLB maintenance instructions targeting EL3 may not invalidate the targeted EL3 TLB entries
21	New	813420	CatB		A57 DCCMVA or DCCSW targeting L2 cache might cause data corruption or deadlock

20 Sep 2013: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
18	New	811671	CatB		A57 issues multiple concurrent DVMOp(Sync) transactions on AMBI 5 CHI interconnect

19	New	812319	CatB		TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2, TLBIIPAS2L instructions might not invalidate targeted IPA cache entry.
45	New	811619	CatC		Return stack erroneous match for not taken indirect branch
46	New	811672	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned
47	New	812170	CatC		Erroneous LR_mon on monitor trap of T32 ACTLR-disabled write accesses
48	New	812171	CatC		Erroneous ESR ISV for ARMv8-added A32/T32 load/store acquire/release exclusive instructions

09 Sep 2013: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
14	New	811120	CatA	Rare	Back-pressure on AMBA 5 CHI TXRSP link can cause data corruption
44	New	809370	CatC		Watchpoint might be incorrectly taken on first half of unaligned ld/st crossing a 64-byte-aligned boundary

28 Aug 2013: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
16	New	808870	CatB		Unconditional VLDM instructions might cause an alignment fault even though the address is aligned
17	New	809371	CatB		Domain fault might be missed when executing specific AT instructions
36	New	806969	CatB	Rare	A57 may send WriteEvict or Evict transaction for cache line that is still valid when using Write-Back No-Allocate memory
42	New	808671	CatC		ETM might not output an Address packet immediately after a Trace Information packet.
43	New	808871	CatC		Double bit ECC error on an invalid L1 data cache line can incorrectly trigger an imprecise abort

Contents

CHAPTER 1. **10**

INTRODUCTION **10**

1.1. Scope of this document **10**

1.2. Categorization of errata **10**

CHAPTER 2. **11**

ERRATA DESCRIPTIONS **11**

2.1. Product Revision Status **11**

2.2. Revisions Affected **11**

2.3. Category A **14**

2.4. Category A (Rare) **14**

811120: Back-pressure on AMBA 5 CHI TXRSP link can cause data corruption 14

2.5. Category B **16**

808870: Unconditional VLDM instructions might cause an alignment fault even though the address is aligned 16

809371: Domain fault might be missed when executing specific AT instructions 17

811671: A57 issues multiple concurrent DVMOp(Sync) transactions on AMBA 5 CHI interconnect 18

812319: TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2, TLBIIPAS2L instructions might not invalidate targeted IPA cache entry..... 19

813419: TLB maintenance instructions targeting EL3 may not invalidate the targeted EL3 TLB entries..... 20

813420: A57 DCCMVA or DCCSW targeting L2 cache might cause data corruption or deadlock..... 21

817169: DSB instruction in processor power down sequence might not drain all required transactions..... 22

817171: AMBA 5 CHI systems using SCLKEN may hang following link deactivation..... 23

826977: L2 Data RAM corruption might occur because of a streaming write to Device-GRE memory which detects a hazard..... 24

829520: Code bounded by indirect conditional branch might corrupt instruction stream..... 25

832075: Cortex-A57 might deadlock when WB exclusive load/store instructions are mixed with device loads.. 27

833471: VMSR FPSCR functional failure or deadlock 28

834921: Code sequence continuously hitting the L1 cache can block snoop..... 29

852523: Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings..... 30

853871: Distributed Virtual Memory operations during hardware flush might cause deadlock or data corruption 31

1319537: Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation 33

1387217: DSB is insufficient to ensure translation table entries being validated are visible to subsequent translations..... 34

1742098: ELR recorded incorrectly on interrupt taken between cryptographic instructions in a sequence..... 35

2.6. Category B (Rare) **36**

806969: A57 may send WriteEvict or Evict transaction for cache line that is still valid when using Write-Back No-Allocate memory 36

826974: Cortex-A57 might violate memory order between an older store and a younger load when executing DMB ALL/ALL instruction..... 37

828024: TLB maintenance operations might not be synchronized by DSB instruction..... 38

834220: Stage 2 translation fault might be incorrectly reported in presence of a stage fault..... 39

836019: Cortex-A57 might deadlock when performing multiple non-overlapping stores with certain memory types and specific timing 40

859972: Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue 41

2.7. Category C **42**

808671: ETM might not output an Address packet immediately after a Trace Information packet. 42

808871: Double bit ECC error on an invalid L1 data cache line can incorrectly trigger an imprecise abort 43

809370: Watchpoint might be incorrectly taken on first half of unaligned ld/st crossing a 64-byte-aligned boundary 44

811619: Return stack erroneous match for not taken indirect branch 45

811672: When a single-bit ECC error occurs in the L2, uncorrected data might be returned. 46

812170: Erroneous LR_mon on monitor trap of T32 ACTLR-disabled write accesses 47

812171: Erroneous ESR ISV for ARMv8-added A32/T32 load/store acquire/release exclusive instructions 48

814669: L2 ECC error might cause a device memory type load to stall until the next timer interrupt 49

814670: A57 DMB barrier does not guarantee observation of the effects of a cache maintenance operation 50

817170: Continuous streaming stores to device memory might block older stores 51

817172: Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption 52

817722: Read after read memory ordering requirements might be violated with specific sequences of exclusive stores and loads 53

821570: Read following a write of a Timer TVAL register might return incorrect value 54

821571: Read or write of ICC_SRE_EL1(NS) might incorrectly generate an undefined exception 55

822224: A57 might deadlock or have data corruption if the page attribute of a current page is remapped by another processor 56

822227: Using unsupported 16K translation granules might cause Cortex-A57 to incorrectly trigger a domain fault 57

822369: HSR.EC might be incorrect for certain accesses when HCR.TGE=1 58

823969: Halting debug event occurring while ITSTATE is non-zero might cause subsequent instructions from the ITR to execute 59

825570: Access to Debug EDSCR.STATUS[5:2] register bits is Read-Write instead of Read-Only 60

826975: Cortex-A57 might livelock because of exclusive load to non-shared memory location with mismatched attributes 61

826978: Cortex-A57 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary 62

828023: Cortex-A57 might violate read-after-read memory ordering when the memory accesses of loads are not single-copy atomic 63

832076: AuxReg field of ID_MMFR0 reports incorrect value 64

832569: Cortex-A57 L1 prefetch works inefficiently for a load stream whose address is mapped to a page size of 64K or larger 65

832769: HSTR.{T7,T8,T15} and HSTR_EL2.{T7,T8,T15} bits incorrectly trap CDP instructions 66

833069: Disabling MMU Translation with CPUACTLR_EL1 "Enable Invalidates of BTB" bit set can cause Invalidate by PA or VA to fail 67

833472: PMU events 0x73 and 0x77 are not incremented for certain instructions 68

834221: Trace packet might be corrupted in certain cases 69

834469: PMU event counters are incorrectly permitted to count events. 70

834528: PMU event BUS_CYCLES might be incorrect in some cases 71

834920: Write of JMCR in EL0 does not generate an undefined exception 72

844270: Syndrome value incorrect for software-induced Virtual Abort exception 73

850774: Cortex-A57 incorrectly allows access to GICv3 common registers in a specific configuration 74

850775: ATB stall from trace subsystem might deadlock the processor 75

851023: Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores 76

852579: Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions 77

852580: Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement 78
852581: Trace Context packet might not be output on a Reset or System Error exception 79
853971: An external data snoop might cause data corruption when an Evict transaction is pending 80
681545: Trace On packets from ETM are not generated in specific conditions around System Error exceptions 81
1185470: Exception packet for return stack match might return incorrect [E1:E0] field 82
1328360: Speculative TLB fills might occur past a DSB instruction 83
1406397: TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0 84
1572968: Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort 85
1742085: Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort 87
1742089: Writes to normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode 88

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-A57 MPCore..

1.1. Scope of this document

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

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

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

1.2. Categorization of errata

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

Table 1 Categorization of errata

Errata Type	Definition
Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A(rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error, or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B(rare)	A significant error, or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

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

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

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

2.2. Revisions Affected

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

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	r1p0	r1p1	r1p2	r1p3
811120	CatA	Rare	Back-pressure on AMBA 5 CHI TXRSP link can cause data corruption	X					
1742098	CatB		ELR recorded incorrectly on interrupt taken between cryptographic instructions in a sequence		X	X	X	X	X
1319537	CatB		Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	X	X	X	X	X	X
1387217	CatB		DSB is insufficient to ensure page table entries being validated are visible to subsequent translations	X	X	X	X	X	X
852523	CatB		Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings	X	X	X	X	X	X
834921	CatB		Code sequence continuously hitting the L1 cache can block snoop	X	X	X	X	X	
833471	CatB		VMSR FPSCR functional failure or deadlock	X	X	X	X	X	
832075	CatB		Cortex-A57 might deadlock when WB exclusive load/store instructions are mixed with device loads	X	X	X	X	X	
829520	CatB		Code bounded by indirect conditional branch might corrupt instruction stream	X	X	X	X	X	
826977	CatB		L2 Data RAM corruption might occur because of a streaming write to Device-GRE memory which detects a hazard	X	X	X	X		

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1	r1p0	r1p1	r1p2	r1p3
817171	CatB		AMBA 5 CHI systems using SCLKEN may hang following link deactivation	X	X				
817169	CatB		DSB instruction in processor power down sequence might not drain all required transactions	X	X				
813420	CatB		A57 DCCMVA or DCCSW targeting L2 cache might cause data corruption or deadlock	X					
813419	CatB		TLB maintenance instructions targeting EL3 may not invalidate the targeted EL3 TLB entries	X					
853871	CatB		Distributed Virtual Memory operations during hardware flush might cause deadlock	X	X	X	X	X	X
812319	CatB		TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2, TLBIIPAS2L instructions might not invalidate targeted IPA cache entry.	X					
811671	CatB		A57 issues multiple concurrent DVMOp(Sync) transactions on AMBA 5 CHI interconnect	X					
809371	CatB		Domain fault might be missed when executing specific AT instructions	X					
808870	CatB		Unconditional VLDM instructions might cause an alignment fault even though the address is aligned	X					
859972	CatB	Rare	Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue	X	X	X	X	X	X
836019	CatB	Rare	Cortex-A57 might deadlock when performing multiple non-overlapping stores with certain memory types and specific timing	X	X	X	X	X	X
834220	CatB	Rare	Stage 2 translation fault might be incorrectly reported in presence of a stage fault	X	X	X	X	X	
828024	CatB	Rare	TLB maintenance operations might not be synchronized by DSB instruction	X	X	X	X		
826974	CatB	Rare	Cortex-A57 might violate memory order between an older store and a younger load when executing DMB ALL/ALL instruction	X	X	X	X		
806969	CatB	Rare	A57 may send WriteEvict or Evict transaction for cache line that is still valid when using Write-Back No-Allocate memory	X					
681545	CatC		Trace On packets from ETM are not generated in specific conditions around System Error exceptions	X	X	X	X	X	X
1185470	CatC		Exception packet for return stack match might return incorrect [E1:E0] field	X	X	X	X	X	X
1328360	CatC		Speculative TLB fills might occur past a DSB Instruction	X	X	X	X	X	X
1406397	CatC		TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0	X	X	X	X	X	X
1572968	CatC		Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort	X	X	X	X	X	X
1742085	CatC		Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort	X	X	X	X	X	X
1742089	CatC		Writes to normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode	X	X	X	X	X	X
853971	CatC		An external data snoop might cause data corruption when an Evict transaction is pending	X	X	X	X	X	X
852581	CatC		Trace Context packet might not be output on a Reset or System Error exception	X	X	X	X	X	X
852580	CatC		Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement	X	X	X	X	X	X

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1	r1p0	r1p1	r1p2	r1p3
852579	CatC		Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions	X	X	X	X	X	X
851023	CatC		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores	X	X	X	X	X	X
850775	CatC		ATB stall from trace subsystem might deadlock the processor	X	X	X	X	X	X
850774	CatC		Cortex-A57 incorrectly allows access to GICv3 common registers in a specific configuration	X	X	X	X	X	X
844270	CatC		Syndrome value incorrect for software-induced Virtual Abort exception	X	X	X	X	X	X
834920	CatC		Write of JMCR in EL0 does not generate an undefined exception	X	X	X	X	X	
834528	CatC		PMU event BUS_CYCLES might be incorrect in some cases	X	X	X	X	X	
834469	CatC		PMU event counters are incorrectly permitted to count events	X	X	X	X	X	
834221	CatC		Trace packet might be corrupted in certain cases	X	X	X	X	X	
833472	CatC		PMU events 0x73 and 0x77 are not incremented for certain instructions	X	X	X	X	X	
833069	CatC		Disabling MMU Translation with CPUACTLR_EL1 "Enable Invalidates of BTB" bit set can cause Invalidate by PA or VA to fail	X	X	X	X	X	
832769	CatC		HSTR.{T7,T8,T15} and HSTR_EL2.{T7,T8,T15} bits incorrectly trap CDP instructions	X	X	X	X	X	
832569	CatC		Cortex-A57 L1 prefetch works inefficiently for a load stream whose address is mapped to a page size of 64K or larger	X	X	X	X	X	
832076	CatC		AuxReg field of ID_MMFR0 reports incorrect value	X	X	X	X	X	
828023	CatC		Cortex-A57 might violate read-after-read memory ordering when the memory accesses of loads are not single-copy atomic	X	X	X	X		
826978	CatC		Cortex-A57 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary	X	X	X	X		
826975	CatC		Cortex-A57 might livelock because of exclusive load to non-shared memory location with mismatched attributes	X	X	X	X		
825570	CatC		Access to Debug EDSCR.STATUS[5:2] register bits is Read-Write instead of Read-Only	X	X	X	X		
823969	CatC		Halting debug event occurring while ITSTATE is non-zero might cause subsequent instructions from the ITR to execute	X	X	X	X		
822369	CatC		HSR.EC might be incorrect for certain accesses when HCR.TGE=1	X	X	X	X		
822227	CatC		Using unsupported 16K translation granules might cause Cortex-A57 to incorrectly trigger a domain fault	X	X	X	X		
822224	CatC		A57 might deadlock or have data corruption if the page attribute of a current page is remapped by another processor	X	X	X	X		
821571	CatC		Read or write of ICC_SRE_EL1(NS) might incorrectly generate an undefined exception	X	X	X			
821570	CatC		Read following a write of a Timer TVAL register might return incorrect value	X	X	X	X	X	X
817722	CatC		Read after read memory ordering requirements might be violated with specific sequences of exclusive stores and loads	X	X				
817172	CatC		Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption	X	X				
817170	CatC		Continuous streaming stores to device memory might block older stores	X	X				
814670	CatC		A57 DMB barrier does not guarantee observation of the effects of a cache maintenance operation	X					

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1	r1p0	r1p1	r1p2	r1p3
814669	CatC		L2 ECC error might cause a device memory type load to stall until the next timer interrupt	X					
812171	CatC		Erroneous ESR ISV for ARMv8-added A32/T32 load/store acquire/release exclusive instructions	X					
812170	CatC		Erroneous LR_mon on monitor trap of T32 ACTLR-disabled write accesses	X					
811672	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned	X					
811619	CatC		Return stack erroneous match for not taken indirect branch	X					
809370	CatC		Watchpoint might be incorrectly taken on first half of unaligned ld/st crossing a 64-byte-aligned boundary	X					
808871	CatC		Double bit ECC error on an invalid L1 data cache line can incorrectly trigger an imprecise abort	X					
808671	CatC		ETM might not output an Address packet immediately after a Trace Information packet.	X					

2.3. Category A

2.4. Category A (Rare)

811120: Back-pressure on AMBA 5 CHI TXRSP link can cause data corruption

Category A Rare
Products Affected: Cortex-A57 MPCore.
Present in: r0p0

Description

A57 configurations that use an AMBA 5 CHI interconnect can encounter data corruption if the system puts back-pressure on the TXRSP link, preventing the core from sending a CompAck response in a timely manner. As a result of this back-pressure, the core might fail to detect a hazard between a new transaction and a pending L2 cache update. This situation can lead to data corruption.

Note: This erratum matches bug #2959 in the ARM internal Jira database.

Configurations Affected

Requires use of AMBA 5 CHI.

Conditions

Due to back-pressure, the A57 TXRSP link does not have sufficient link-layer credits to allow immediate issue of CompAck following the last data beat of a ReadClean, ReadShared, ReadUnique, or CleanUnique transaction.

Implications

If the above scenario is encountered, data corruption can occur.

Workaround

There is no workaround.

2.5. Category B

808870: Unconditional VLDM instructions might cause an alignment fault even though the address is aligned

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

An unconditional Vector Load Multiple (VLDM) instruction that loads multiple 64-bit registers when SCTLR.A (Alignment check enable) is set to 1 can generate more alignment faults than required. The core should only generate an alignment fault when the address is not 4-byte aligned (address[1:0]=0x0), but due to this erratum it will generate an alignment fault when the address is not 8-byte aligned (address[2:0]=0x0).

Note: This erratum matches bug #2858 in the ARM internal Jira database.

Configurations Affected

All

Conditions

1) Unconditional VLDM in AArch32 state that specifies a <list> of 64-bit registers
2) Value of base register Rn that is 4-byte aligned but not 8-byte aligned (address[2:0]=0x4)
3) System Control Register (EL1/EL2/EL3) with A=1

Implications

An alignment fault will be generated when the base address is not 8-byte aligned even if it is 4-byte aligned.

Workaround

To avoid this erratum, if an alignment fault occurs, check DFAR to see if the fault address is 4-byte aligned (DFAR[1:0] = 0x0). If it is, emulate the faulting instruction in software using a conditional VLDM that is guaranteed to pass the condition code check.

809371: Domain fault might be missed when executing specific AT instructions

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

When certain address translation (AT) instructions are executed in AArch64 EL3 targeting secure EL1 (SCR.NS=0) with the VMSAv7 translation table format (TCR_EL1.EAE=0), the core will not trigger a domain fault when encountered as defined by DACR_NS[31:0].

Note: This erratum matches bug #2900 in the ARM internal Jira database.

Configurations Affected

Requires use of AArch64.

Conditions

1) Currently in EL3 which is using AArch64.
2) Execute certain address translation (AT) instructions targeting the secure EL1 (ATS1E1R, ATS1E1W, ATS1E0R, ATS1E0W, ATS12E1R, ATS12E1W, ATS12E0R, ATS12E0W).
3) The secure EL1 is using VMSAv7 translation system (TCR_EL1.EAE=0).
4) The secure EL1 is using the domain.
5) The memory access should trigger a domain fault as defined by DACR_NS[31:0].

Implications

If the conditions above are met, the core will not trigger a domain fault as required.

Workaround

If the above conditions apply and there is a possibility of a domain fault, EL3 should use software to walk the secure EL1 page tables.

811671: A57 issues multiple concurrent DVMOp(Sync) transactions on AMBA 5 CHI interconnect

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

Dual and quad-core A57 configurations that use an AMBA 5 CHI interconnect can cause a system deadlock by issuing more than one concurrent DVMOp(Sync) transactions.

Each CPU can issue a single DSB instruction to the L2 cache system, each of which can issue a DVMOp(Sync) transaction to the interconnect. AMBA 5 CHI architecture mandates that MN must have sufficient resources to accept all DVMOp(Sync) transactions in the system and still have

resources to accept at least two non-DVMOp(Sync) transactions with certain restrictions on their usage. Depending on the particulars of the MN implementation and a given number of RN-Fs in the system, a single RN-F issuing more than one concurrent DVMOp(Sync) transactions can violate this agreement and can lead to system deadlock due to a resource dependency chain. Note: The CCN-504 configuration allows for a maximum of four RN-F devices (for example an A57 cluster) and has an MN with a 16-entry transaction buffer. This MN buffer size allows for a maximum of 14 concurrent DVMOp(Sync) transactions plus two non-DVMOp(Sync) transactions. Therefore, systems using CCN-504 with fewer than 14 total sources of DVMOp(Sync) (where each A57 cluster can contribute one DVMOp(Sync) from each core) are not affected by this erratum. For example, a system consisting of four dual-core A57 clusters will not be affected by this erratum.

Note: This erratum matches bug #2927 in the ARM internal Jira database.

Configurations Affected

Requires use of AMBA 5 CHI in a multi-core system. If using CCN-504, only configurations with more than 14 total cores across all A57 MP Core clusters are affected.

Conditions

1) Two or more CPUs perform DSB instructions.
2) Each DSB instruction causes an EOBarrier and a DVMOp(Sync) on the AMBA 5 CHI RN-F interface
3) Multiple DVMOp(Sync) transactions from a single RN-F can cause a resource dependency chain through the MN that leads to a deadlock.

Implications

If the above conditions occur, the system might deadlock.

Workaround

1) Disable the broadcast of DVM transactions by setting L2ACTLR[8]. Note that this disables the broadcast of all DVM transactions, including TLBI, ICI and BPI, from one cluster to another and also disables the broadcast of cache maintenance operations for non-shareable memory to an L3 cache, or,
2) Prevent multiple DVMOp(Sync) transactions by allowing only one CPU at a time to issue a 'heavy' DSB operation to the L2 Cache.

The workaround (1 OR 2) relies on logic within the core that creates two flavors of DSB transactions to the L2 cache system. This logic detects whether an operation that is synchronized by a DVMOp(Sync) has occurred since the last DSB instruction was executed. If such operation has occurred since the last DSB, the core will indicate that a DSB is 'heavy' and this DSB will cause an EOBarrier AND a DVMOp(Sync). If no such operation has occurred since the last DSB, the core will indicate that the DSB is 'light' and will cause only an EOBarrier. The operations that force a 'heavy' DSB include TLBI*, IC* and BPI*. Since these operations are only allowed in EL1 or higher (except IC IVAU in AArch64 mode which can be disabled for EL0 using SCTLR.EL1.UCI), EL0 DSBs will never cause DVMOp(Sync). EL1 and higher can implement a locking scheme to allow only one CPU within a cluster to perform the operations that cause DSBs to become 'heavy' and guarantee that a DSB will always follow such operations before switching back to EL0.

812319: TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2, TLBIIPAS2L instructions might not invalidate targeted IPA cache entry.

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

The new ARMv8 A32 and T32 TLBI maintenance instructions to caches of Stage 2 caches by IPA will not invalidate the targeted IPA cache entry if the supplied IPA[39:32] is non-zero. This includes TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2 and TLBIIPAS2L instructions.

Note: This erratum matches bug #2970 in the ARM internal Jira database.

Configurations Affected

All processor configurations are affected.

Conditions

1) A32 and T32 TLBIIPAS2IS, TLBIIPAS2LIS, TLBIIPAS2, TLBIIPAS2L instructions used by hypervisor or secure EL3.
2) The IPA supplied by the instruction has non-zero IPA[39:32].

Implications

ARMv8 A32 and T32 TLBI maintenance instruction to caches of Stage 2 caches by IPA cannot be used with non-zero IPA[39:32]. However, existing ARMv7 TLBI VMALLEL1 instructions function properly.

Workaround

Hypervisor or secure EL3 can use TLBI VMALLEL1 instructions for TLBI maintenance.

813419: TLB maintenance instructions targeting EL3 may not invalidate the targeted EL3 TLB entries

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

Following the recommended ARMARM TLB maintenance procedure, {TLBI, DSB} is issued after the page table is modified. When the corresponding TLBI*EL3 and DSB instructions are received by the CPU, the CPU will correctly invalidate all targeted EL3 TLB entries in L1DTLB. However, if the CPU has an outstanding EL3 table walk request from the L1DTLB, the TLBI*EL3 and DSB will only guarantee that the table walk has completed and been written into L1DTLB, but will not invalidate this entry which could retain the old translation. The affected TLBI instructions include all TLBI instructions targeting EL3, including TLBI VAE3, TLBI VALE3, TLBI VAE3IS, TLBI VALE3IS, TLBI ALLE3 and TLBI ALLE3IS.

Note: This erratum matches bug #3018 in the ARM internal Jira database.

Configurations

Systems implementing secure AArch64 EL3.

Conditions

1) A CPU (cpuA, any ARMv8 AArch64 core) modifies a page table and executes a TLBI targeting EL3 translations.
2) An A57 L1DTLB (on cpuA or another A57 cpu) receives the TLBI command.
3) That L1DTLB has a translation request outstanding for one of the pages to be invalidated when the TLBI command is received.
4) That translation request returns a stale translation.

Implications

This bug will only affect secure AArch64 EL3. If the above conditions occur, the CPU will not invalidate the targeted EL3 TLB entries and incorrect translations might occur.

Workaround

The software workaround is to add another pair of TLBI*EL3 and DSB. The completion of the first pair of {TLBI*EL3, DSB} will guarantee that the outstanding table walk request finishes and the translation has been written into the L1DTLB. The additional pair of TLBI*EL3 and DSB will guarantee that the old translation in L1DTLB is invalidated. Note that the TLB entry with the old translation could be invalidated because of the capacity eviction and therefore initiate a new table walk request. The new table walk request will get new translation provided the recommended ARMARM TLB maintenance procedure is followed.

813420: A57 DCCMVA or DCCSW targeting L2 cache might cause data corruption or deadlock

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

A DCCMVAC, DCCMVAU, or DCCSW targeting the L2 cache that hits a dirty line creates a WriteClean transaction. If a snoop to the same cache line address as the WriteClean collides with the Fill/Evict Queue entry of the WriteClean and hits in the L2 cache, the resulting L2 cache access may cause 1) data corruption in an unrelated Fill/Evict Queue entry, or 2) early de-allocation of the Fill/Evict Queue entry of a second snoop to the same cache line address. The early de-allocation case might cause deadlock on the RXRSP channel.

Note: This erratum matches bugs #3013 and #3014 in the ARM internal Jira database.

Configurations

AMBA 5 CHI configurations.

Conditions

1) DCCMVAC, DCCMVAU, or DCCSW targeting L2 cache that hits in the L2 cache to a line in a dirty state which creates a WriteClean transaction and changes the L2 cache to a clean state.
2) An external snoop to the cache line address of the WriteClean.

Implications

If the above conditions are met, data corruption or deadlock might occur.

Workaround

Set CPUACTLR[44] to execute data cache clean operations as data cache clean and invalidate. Note that in most systems CPUACTLR is only accessible in EL3 and therefore needs to be written by firmware in secure state.

817169: DSB instruction in processor power down sequence might not drain all required transactions

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1

Description

When powering down an individual core in a Cortext-A57 MPCore processor it is possible that a DSB or DVM operation from another processor might not be completely flushed before power is removed. If the processor is powered down before the DSB or DVM operation completes the system might deadlock.

When executing a processor power-down sequence as specified in the A57 TRM, the CPUECTLR.SMPEN bit is cleared, followed by an ISB and then a DSB to drain all required transactions from other processors before allowing the current processor to power down. A performance optimization was added that allowed DSB instructions to execute more quickly if no DVM operations (TLB, branch predictor, or instruction cache maintenance operations) have been executed since the previous DSB. This faster version of the DSB does not perform all the actions required to correctly drain requests from other cores as required by the power-down sequence.

Note: This erratum matches bug #3078 in the ARM internal Jira database.

Configurations Affected

All configurations

Conditions

1) A DSB or DVM operation is executed on a processor (cpuA).
2) No DVM operation (TLB, branch predictor, or instruction cache maintenance operations) has been executed on cpuB since the last DSB instruction on a different processor (cpuB).
3) CPUECTLR.SMPEN bit is set to 0 followed by ISB/DSB on cpuB which is being powered down.
4) cpuB is powered down.
5) The DSB or DVM operation is sent to cpuB and is dropped.

Implications

If the above conditions occur, the DSB or DVM operation will never complete and the system will stall until the next reset.

Whether or not the deadlock is seen will depend heavily on the timing of the power-down sequence. If there is a long period of time between clearing the CPUECTLR.SMPEN and the actual removal of power from the processor, it is very likely any in-flight operations from other cores will have time to complete.

Workaround

After clearing the CPUECTLR.SMPEN bit, and after the required ISB, execute a translation look-aside buffer entry invalidate instruction (TLBIVA) before the required DSB. Any TLB address can be referenced by the instruction. This TLBIVA will cause the DSB execution to drain all appropriate transactions from other processors and will prevent the deadlock.

817171: AMBA 5 CHI systems using SCLKEN may hang following link deactivation

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1

Description

A57 systems configured to have a CHI interface and regional clock gating and using the SCLKEN input to implement integer ratioed CHI interface can underflow link credit counters during link deactivation.

Note: This erratum matches bug #3086 in the ARM internal Jira database.

Configurations affected

A57 systems with regional clock gating enabled and an AMBA 5 CHI interconnect using SCKLEN for an integer ratioed CHI interface.

Conditions

The CHI link is deactivated as a result of the cluster entering STANDBYWFIL2 or the deassertion of the RXLINKACTIVEREQ input.

Implications

If the above condition occurs, the TXREQFLITV, TXRSPFLITV, and TXDATFLITV outputs might remain asserted following CHI link deactivation. This condition might cause underflow of the link credit counters and a system deadlock.

Workaround

Set L2ACTLR[26] to 1'b1 to disable L2 regional clock gating.

826977: L2 Data RAM corruption might occur because of a streaming write to Device-GRE memory which detects a hazard

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

If a streaming write is issued from a processor to a physical address mapped to Device-GRE memory and the transaction detects a hazard with an older load to the same 64-byte physical address within the Level2 memory system, the streaming write is replayed. If an eviction from the processor immediately follows the streaming write down the L2 pipeline, it might result in the eviction using a portion of data from the streaming write to update the line in the L2 cache associated with the eviction, thereby corrupting the contents of the line in the L2 cache. This only applies to systems which support the minimum L2 tag latency of 2 cycles. For these systems, there is a one cycle window in which the eviction can advance to a point in the pipeline to extract the data from a series of data stage registers, which might contain data associated with the replayed streaming write. If a write request is replayed, then the contents of the data stage registers are flushed. However, in systems supporting the minimum 2-cycle tag

latency, the eviction might extract the data from the data stage registers prior to the eviction's data overwriting the flushed streaming write's data. This issue requires a streaming write. Only streaming writes to Device-GRE memory can cause this issue because other normal memory types are ordered by the processor. SO/DEV memory transactions do not apply, because they will not be issued as a streaming write.

Note: This erratum matches bug #3197 in the ARM internal Jira database.

Configurations affected

Systems which have an overall L2 Tag RAM latency = 2 cycles, which implies the following:

- Tag slice = 0
- L2CTLR[9] = 0 - no tag setup cycle
- L2CTLR[8:6] = 0 or 1 - programmed tag latency = 2

In addition to the minimum overall L2 Tag RAM latency listed above, the system must have streaming writes enabled and support Device-GRE memory attributes.

Conditions

1) Processor(X) Device-GRE streaming write is replayed because of an address hazard with an older Device-GRE load, which flushes staged streaming write data.
2) Only critical quadword of streaming writes arbitrates for the L2 pipeline. Non-critical quadword accesses are blocked because of conflicts.
3) Processor(X) L1 data cache eviction issues down L2 pipeline immediately after the Device-GRE streaming write.
4) Processor(X) L1 data cache eviction uses data associated with streaming write before eviction data can overwrite staged streaming write data.
5) L2 data cache line corresponding to the L1 data cache eviction is corrupted.

Implications

If the above conditions exist then the contents of the data RAM associated with the L1 data cache eviction might be corrupted. Corruption of the L2 data cache will only occur for dirty L1 data cache evictions.

Workaround

Treat Device-GRE/Device-nGRE as Device-nGnRE by setting CPUACTLR_EL1[54]=1.

829520: Code bounded by indirect conditional branch might corrupt instruction stream

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

An indirect conditional branch will have its direction predicted by the main and micro-predictor, but the indirect predictor will only be used if more than one target is detected for the branch. The micro-predictor cannot be used to predict polymorphic (more than one target) indirect branches. Upon detection of a polymorphic branch, in certain cases, the micro-predictor might not be properly inhibited.

Note: This erratum matches bug #3229 in the ARM internal Jira database.

Configurations affected

Code that is running in the A32 or T32 instruction sets.

Conditions

1) A sequence of instructions that spans three 128-bit aligned memory regions is terminated by a conditional indirect branch instruction. These instructions include LDR PC, MOV PC, and BX, but do not include instructions that can be considered subroutine returns.
   - Note that this class of instructions does not exist in the A64 instruction set.
2) The sequence must not contain any other taken branches. The only branches permitted in the sequence are branches that are never taken.
3) The indirect conditional branch targets the address at the head of the sequence, then changes the target address.
4) The indirect conditional branch becomes not taken after the target address has changed.
5) The not taken indirect conditional branch is correctly predicted by the branch predictor.

If these conditions are met, then the branch predictor will redirect the instruction fetch to the target of the indirect conditional branch (i.e. the head of the sequence of instructions) when the branch is not taken, causing a mismatch between the reported PC and the instruction stream.

A candidate code that meets these criteria is shown below:

```
        mov       r0, #12
        adr       r9, target_table
target2
        nop
target1
        mov       r1, #1
        mov       r2, #2

        add       r3, r1, r2
        add       r4, r2, r2
        add       r5, r4, r3
        subs      r0, r0, #1

        ldr       r10, [r9], #4
        bxne      r10
fall_thru
        nop
        nop
```

The erratum manifests when 'target1' is the address targeted by the 'bxne r10' instruction, and at a later time 'target2' becomes the targeted address. The symptom is the instructions at 'target1' appear to be resident at the 'fall_thru' address when the branch is not taken. Note that the branch predictor must also be in a state to correctly predict the not taken branch in order for the erratum to occur.

Implications

The 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.

Workaround

The CPUACTLR_EL1 Disable Indirect Predictor bit will prevent this erratum from occurring. Set CPUACTLR_EL1[4] = 1'b1 to disable the Indirect Predictor. This might cause some performance degradation depending on the application.

832075: Cortex-A57 might deadlock when WB exclusive load/store instructions are mixed with device loads

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

Cortex-A57 might deadlock when exclusive load/store instructions to Write-Back memory are mixed with Device loads.

Note: This erratum matches bug #3247 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

This erratum requires a particular sequence of code along with some specific internal timing:

1) One or more store instructions to a Write-Back memory location, that has the same L1D set address (physical address [13:6]) as the exclusive load and exclusive store instructions identified in 2) below.
2) A pair of exclusive load and exclusive store instructions. The exclusive store is to a Write-Back memory location.
3) A branch instruction has a source register which is the same as the destination register of the older exclusive store identified in 2) above.
4) Following the branch instruction, there are six or more loads to Device memory locations.

Implications

Cortex-A57 might deadlock when Write-Back exclusive load/store instructions are mixed with Device loads. Note that an interrupt will break the deadlock.

Workaround

The preferred workaround is to have one of the first five Device loads that follow the load/store exclusive pair use Acquire semantics (Load-Acquire).

If this is not possible, one of the following workarounds will prevent this erratum from happening. They are listed in order of increasing impact on performance:

- Insert a CLREX instruction after the original exclusive store.\
- Replace the exclusive store with an exclusive release store.
- Insert a DMB LD/ALL or DMB ALL/ALL instruction between the exclusive load/store pair and the Device loads.

833471: VMSR FPSCR functional failure or deadlock

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

In rare circumstances involving particular T32 code sequences with conditional execution and involving specific internal execution delays or asynchronous exceptions, VMSR FPSCR might fail to update FPSCR correctly or deadlock might result.

Note: This erratum matches bug #3270 in the ARM internal JIRA database.

Configurations affected

Requires use of AArch32.

Conditions

In AArch32 state, the following T32 code sequence is executed:

```
 IT {<x>{<y><{z}}} <firstcond>
     {IT block instructions 1-3 if present}
     RFE   ; (RFE is last instruction in IT block & RFE fails its condition code check)
   VMSR  FPSCR, Rx
```

1) Considerable internal delays between fetch or execution of the T32 RFE as last instruction in IT block that fails its condition code check and execution of its immediately preceding instruction.
2) The instruction immediately preceding the T32 RFE as last instruction in IT block that fails its condition code check is an ISB that passes its condition code check.
3) An asynchronous exception is taken on the T32 RFE as last instruction in IT block that fails its condition code check.

Implications

If conditions 1 AND (2a OR 2b OR 2c) are met, the VMSR FPSCR might fail to update FPSCR correctly or deadlock might result. Note that this condition is currently only seen in contrived (non-useful) validation code. The erratum has not been seen in the real applications. These conditions are not possible in user mode because of the use of the RFE instruction.

Workaround

Set CPUACTLR[38] to 1, which forces FPSCR write flush. Note that in some cases where a flush is unnecessary this could impact performance.

834921: Code sequence continuously hitting the L1 cache can block snoop

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

If a sequence of memory operations is executed such that a single tag bank or a single data bank of the L1 data cache is accessed every cycle, and a store instruction is executed periodically (at least once every 32 cycles), a snoop to a physical address that is unrelated to the load and store instructions might be blocked until there is a single cycle where a load is not accessing that tag or data bank. This might block the snoop until the current polling loop finishes or until the next interrupt or other exception.

Note: This erratum matches bug #3302 in the ARM internal Jira database.

Configurations affected

All configurations are affected.

Conditions

1) Continuous stream of load instructions hitting in the L1 data cache every cycle.
2) The load physical addresses (PA) must meet the following criteria:
   - PA[7:6] for all loads are the same (same tag bank) OR
   - PA[5:4] for all loads are the same (same data bank)
3) A store that hits the L1 data cache must occur at least once every 32 cycles.
4) A snoop (from the interconnect or another A57 core) hits the L2 cache while L2 cache is waiting for the eviction of this cacheline from L1 data cache.
5) A software dependency that means the continuous stream of loads continues until the snoop completes.

Implications

If all conditions are met, the snoop could be held off until the next interrupt. This would only happen if the processor is executing an unusual polling loop containing a store waiting for a cacheable memory location to be updated. To hit the condition, the polling loop would need to contain at least two load instructions and a store instruction, all hitting the L1 data cache. The conditions cannot be met if the loop contains any load acquire, load exclusive, WFE, DMB, or DSB instructions.

Workaround

Untrusted or user code must be run with periodic timer interrupts, that will prevent the erratum from causing a deadlock. Alternatively, if a software polling loop is found to be hitting this erratum, simplifying the polling loop to contain only the loads that are checking the required conditions and delaying any store instructions until after the conditions have been met will avoid the erratum.

852523: Writes to DACR32_EL2 in AArch64 state might not have desired effect on domain settings

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

AArch32 memory domain access permissions are controlled by the DACR register. This register is accessible in AArch64 state by reading or writing the DACR32_EL2 register. If a Cortex-A57 processor attempts to write the DACR in AArch64 state using the DACR32_EL2 register, the updated register settings might not be seen by the processor.

Note: This erratum corresponds to issue #3326 in the ARM internal tracking system.

Configurations affected

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

Conditions

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

Implications

If the above conditions are met, the Cortex-A57 processor might not properly use the updated DACR written by the hypervisor. This might result in spurious domain faults in the guest O/S, or possibly allow EL0 processes in the guest O/S access to the operating system privileged data or code.

Workaround

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

853871: Distributed Virtual Memory operations during hardware flush might cause deadlock or data corruption

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

During execution of L2 hardware cache flush, as part of the powerdown sequence, Distributed Virtual Memory (DVM) operations received by the Cortex-A57 processor might be dropped leading to a deadlock or data corruption.

Note: This erratum corresponds to issue #3332 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) L2 hardware cache flush is initiated following the powerdown sequence in the Cortex-A57 Technical Reference Manual by asserting L2FLUSHREQ.
2) During the L2 hardware cache flush sequence, the Cortex-A57 processor receives external DVM operations.
3) The L2 Fill Evict Queue (FEQ) becomes completely full with L2 hardware cache flush transactions or external DVM operations.
4) Very specific timing, so an L2 hardware cache flush transaction is allocated to a particular FEQ entry in the same cycle that another FEQ entry is deallocated.
5) DVM operation is later allocated to the last entry in the FEQ, causing the FEQ to be completely full.

Implications

If the above conditions are met, the Cortex-A57 processor drops the external DVM operation in step 5). This might cause data corruption for subsequent snoops and prevents the powerdown sequence from completing, because the Cortex-A57 processor does not respond to all the external DVM operations, and the system will deadlock.

Workaround

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

1) The preferred workaround involves the modification of the power management software running on the SCP using the L2 hardware cache flush mechanism as described in the ARM Cortex-A57 MPCore Processor Technical Reference Manual (rev1.3) section "Processor powerdown with system driven L2 flush". This mechanism involves using an implementation defined mechanism within the SoC to drive the signals L2FLUSHREQ and either ACINACTM or SINACT. In order to work around this erratum, while still using this mechanism, the power management software must be able to disable DVM operations before the SoC assertion of L2FLUSHREQ (step 6 of documented sequence) separately from disabling snoop requests (which is typically done by the management software between steps 8 and 9 of the documented sequence by writing to a register within the SoC level interconnect). For the ARM implemented interconnects, the following approaches are used:
    - For CCI-400/500/550 based ACE interconnect systems, the power management software must program the Snoop Control Register (snoop_ctrl) to disable DVM operations to the Cortex-A57 processor and poll the Status Register to confirm changes to the Snoop Control Register have taken effect, just before step 6, where the SoC asserts L2FLUSHREQ. After step 8, where L2FLUSHDONE is asserted by the Cortex-A57 , program the Snoop Control Register to disable snoop requests and poll the Status Register to confirm changes have taken effect before asserting ACINACTM (this operation is required even without the errata

workaround, but would also have including the disabling of DVM operations at this point). Please refer to the appropriate CCI documentation for disabling snoops and DVM operations.

- For CCN-based AMBA 5 CHI interconnect systems, the power management software must write to the DVM Domain Control Clear Register (DDCR_Clear) to disable DVM operations to the Cortex-A57 processor and poll the DVM Domain Control Register (DDCR) to confirm changes to the DDCR have taken effect, just before step 6 where the SoC asserts L2FLUSHREQ. After step 8, where L2FLUSHDONE is asserted by the Cortex-A57, program the Snoop Domain Control Clear Register (SDCR_clear) to disable snoops and poll the Snoop Domain Control Register (SDCR) to confirm changes to the SDCR have taken effect (this operation is required even without the errata workaround but would also have including the disabling of DVMs at this point). Please refer to the appropriate CCN documentation for disabling snoops and DVM operations.

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

2) If the SoC is such that DVM operations cannot be disabled independently from disabling snoops by power management software, then the caches must be cleaned and invalidated by software running the Cortex-A57 core as part of the power management software, following the steps in the Cortex-A57 MPCore Technical Reference Manual section “Processor powerdown without system driven L2 flush” using data or unified cache line clean and invalidate by set/way instructions (DCCISW) to clean and invalidate all data from the L2 data cache.

1319537: Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3342 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

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

Implications

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

Workaround

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

1387217: DSB is insufficient to ensure translation table entries being validated are visible to subsequent translations

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

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

Note: This erratum corresponds to issue #3344 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

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

Implications

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

Workaround

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

1742098: ELR recorded incorrectly on interrupt taken between cryptographic instructions in a sequence

Category B

Products Affected: Cortex-A57 MPCore.

Present in: r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

or

AESD Qy, Qx
AESIMC Qy, Qy

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

Note: This erratum corresponds to issue #3351 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations which implement the cryptography extension.

Conditions

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

Implications

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

Workaround

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

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

Other AES modes that match condition 2 can workaround the issue in a similar manner.

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

2.6. Category B (Rare)

806969: A57 may send WriteEvict or Evict transaction for cache line that is still valid when using Write-Back No-Allocate memory

Category B Rare

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

Based on memory page attributes from the translation tables, type of load, and dynamic mechanisms, the core may treat a given load as write-back no-allocate. For such loads, the load-store unit can use a fill buffer allocated by an earlier load to satisfy subsequent loads to the same cache line. This means that the fill buffer can remain valid for some amount of time after data is initially returned from the coherent interconnect. During this time, any coherent activity to this cache line must be seen by the core through the external snoop request interface (AC channel in AMBA 4 ACE configurations and RXSNP channel in AMBA 5 CHI configurations). When the core processes a shareable write-back no-allocate load that misses in the L1 and L2 caches, it sends a ReadClean transaction on the coherent interconnect. When all the data is returned, based on L2ACTLR configuration bits and the coherent state of the cache line, the core may send a WriteEvict or Evict transaction. Upon completion, a WriteEvict and Evict indicate to the system that the core no longer maintains a copy of the cache line. This can cause a system which maintains a snoop filter to no longer send snoops to this cluster, which in turn causes the core to violate coherency and ordering rules.

Note: This erratum matches bug #2802 in the ARM internal Jira database.

Configurations Affected

Both AMBA 4 ACE and AMBA 5 CHI configurations in systems that include a snoop filter.

Conditions

1) BROADCASTINNER and/or BROADCASTOUTER inputs asserted (i.e. cluster is hardware coherent)
2) The processor is connected to a system that filters snoops (e.g. CCN-504)
3) Evict and/or WriteEvict transactions are enabled (L2ACTLR[14]==1 or L2ACTLR[3]==0)
4) A load that is treated as write-back no-allocate by the load-store unit is performed (memory mapped write-back no-allocate as defined by the translation tables, a non-temporal load, or a load to a L1 data cache set that already has two misses outstanding)
5) An invalidating snoop (SnpUnique, SnpCleanInvalid, SnpMakeInvalid) from another coherent master targeting the cache line being loaded

Implications

If the above conditions occur, the core might not receive an invalidating snoop, causing a loss of coherency and ordering, which may lead to data corruption.

Workaround

To avoid this erratum, do one of the following:

1) Disable load-store use of write-back no-allocate memory type by setting CPUACTLR[49] to 1

2) Disable WriteEvict and Evict transactions by setting L2ACTLR[3] to 1 and L2ACTLR[14] to 0

The first workaround is the preferred solution since the second workaround will reduce system performance.

826974: Cortex-A57 might violate memory order between an older store and a younger load when executing DMB ALL/ALL instruction

Category B Rare

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Cortex-A57 might violate the memory ordering requirements between an older store and a younger load which are separated by a DMB ALL/ALL instruction when the younger load requires memory accesses to two cachelines.

Note: This erratum matches bug #3202 in the ARM internal Jira database.

Configurations affected

All configurations are affected.

Conditions

1) DMB ALL/ALL instruction is used to enforce memory ordering between an older store and a younger load.
2) The younger load requires access to two cachelines. The memory location being accessed by the load is used for synchronization.
3) When the first half of the load gets the data from the L1 data cache, there is a precise abort due to a younger store. Also the second half of the load does not get its data and therefore the load will replay.
4) During the time window between the first half of the load getting the data from L1 data cache and the replay of the load, the cacheline is evicted or snooped out of the L1 data cache and is modified with newer data by another master.
5) The older store is then written into cache and is observed by another processor. This is after the younger load is observed by other processors and thus violates the memory ordering as imposed by DMB ALL/ALL.

Implications

If the load being split across the cacheline boundary is used for any algorithm which relies on the memory ordering between the store and the load as being enforced by DMB ALL/ALL instruction, such as Dekker's algorithm, the erratum might break the algorithm. In case of Dekker's algorithm, it requires that the conditions listed above occur on the loads of both processors.

Workaround

Set CPUACTLR_EL1[59]=1 to disable speculative load execution ahead of a DMB.

828024: TLB maintenance operations might not be synchronized by DSB instruction

Category B Rare

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

A TLB invalidate instruction followed by a DSB instruction to ensure its completion, should remove all uses of the old translation from the system. On Cortex-A57 the DSB instruction might complete before some store memory transactions to the invalidated translation are globally observed. This affects store memory transactions with write-back-no-allocate (due to page attributes or streaming stores), write-through, non-cacheable, device, or strongly-ordered memory attributes. This does not affect write-back-read-write-allocate stores.

This might lead to data corruption if the software attempts to access or allows access to the physical memory of the invalidated or moved region before the stores have completed.

Note: This erratum matches bugs #3219 in the ARM internal JIRA database.

Configurations Affected

Multi-cluster systems that utilize distributed virtual memory (DVM) transactions.

Conditions

1) Software modifies the translation tables to invalidate, restrict the permissions of, or change the physical address of a page or block.
2) Software executes a TLB invalidate instruction to invalidate any TLB entries holding the previous translation.
3) Software executes a DSB instruction to ensure completion of the TLB invalidate instruction and global observation of any memory transactions that depended on the previous translation.

Implications

When software changes the translation tables and invalidates a modified TLB entry, outstanding stores to the old translation might not be globally observed. This might lead to data corruption if the physical memory of the invalidated memory region is accessed before the memory requests that used the old translation are complete.

Workaround

By following the workaround below, this erratum will not affect cacheable operations, therefore it becomes a Category C erratum that only affects store transactions with write-through, non-cacheable, device, or strongly-ordered memory attributes. Note that disabling write streaming will have an adverse effect on the performance of memset and memcpy operations.

1) Set CPUACTLR_EL1[49] = 1 // disable non-allocate hint of write-back-no-allocate memory type
2) Set CPUACTLR_EL1[26:25] = 2'b11 // disable write streaming no L1-allocate threshold
3) Set CPUACTLR_EL1[28:27] = 2'b11 // disable write streaming no-allocate threshold

834220: Stage 2 translation fault might be incorrectly reported in presence of a stage fault

Category B Rare

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

Cortex-A57 might report a stage 2 translation fault instead of a stage 1 permission fault or device memory alignment fault as required by the ARM architecture for a load crossing the page boundary when there is a stage 1 permission or device memory alignment fault and a stage 2 translation fault.

Note: This erratum matches bug #3292 in the ARM internal JIRA database.

Configurations affected

Systems running with a hypervisor.

Conditions

1) A load spanning across the page boundary.
2) The translation of the first page returns a stage 1 permission fault or a device memory alignment fault on the final IPA and a stage 2 translation fault. This translation with fault is saved in a side fault information structure.
3) The load does not report the fault immediately when there are other microarchitecture hazards, such as a committed DMB. Meanwhile, Cortex-A57 L1 DTLB initiates the TLB miss request for the second page of the load.

Under rare timing conditions, Cortex-A57 might report the stage 2 translation fault instead of the stage 1 permission or device memory alignment fault as required by the architecture.

The erratum does not occur if the stage 1 fault is a general alignment fault.

Implications

When a system is running with a hypervisor and there is a stage 1 permission or device memory alignment fault and a stage 2 translation fault, Cortex-A57 might report the stage 2 translation fault instead of the stage 1 fault as required by the ARM architecture. If the hypervisor fixes the stage 2 translation fault and the same instruction is replayed, Cortex-A57 reports the stage 1 fault.

Workaround

The workaround requires the hypervisor to check for any stage 1 permission fault or device memory alignment fault when taking a stage 2 translation fault.

This can be implemented by the following:

- When ESR_EL2 reports a stage 2 read fault taken from ELx, an AT ATS1ExR instruction can be used with the reported FAR_EL1 value to determine whether there is also a stage 1 fault on the access.
- If PAR_EL1 indicates a permission fault or device memory alignment fault at stage 1, then the reported stage 2 translation fault might be invalid and the hypervisor should return to the faulting instruction.

836019: Cortex-A57 might deadlock when performing multiple non-overlapping stores with certain memory types and specific timing

Category B Rare

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

The processor performs a series of non-overlapping stores to L2 cache line (A), targeting memory mapped as Non-cacheable (NC), Device-nGRE, Device-GRE or Write-back no allocate (WBNA). These writes are followed by another transaction B to the same physical cache line (A). In rare cases, when the stores to cache line A are sent out-of-order to the interconnect, the processor might incorrectly hazard a newer unrelated transaction, resulting in deadlock.

Note: This erratum matches bugs #3306 in the ARM internal tracking system.

Configurations affected

Systems using AMBI 5 CHI and configured with an L2 arbitration slice are affected.

Conditions

1) A series of non-overlapping stores are performed in close succession to L2 cache line (A), targeting memory that is mapped as either NC, Device-nGRE, Device-GRE or WBNA.
2) The stores are immediately followed by another transaction (B), targeting the same L2 cache line A. The transaction B would then detect an address hazard, against any pending stores to cache line A, and be tagged with a hazard against the youngest store.
3) The stores are sent out-of-order to the interconnect, with the youngest entry being processed before some of the older pending entries. This causes the hazard against the oldest entry to be cleared.
4) An instruction fetch or tablewalk request occurs in the cycle after the hazard clears.
5) The processor performs another memory transaction (C).
6) Conditions 1-5 occur with very specific timing

Implications

If the above conditions occur, transaction B might detect a new hazard against the next youngest store. This new hazard might be incorrectly passed on to transaction C, even though the store in question has already been completed, resulting in deadlock.

Workaround

Set L2ACTLR[7] to enable hazard detection timeout. This will force the processor to periodically re-evaluate hazards, which would clear the false hazard resulting from this erratum. There are no performance implications to setting this bit.

859972: Speculative instruction prefetch to Execute-never (XN) memory could cause deadlock or data integrity issue

Category B Rare

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

ARM architecture prohibits speculative instruction fetches to addresses that are marked as Execute-never (XN) in the page tables. However, where a Normal Write-Back Cacheable page with execute permission precedes a page with Execute-never (XN) permission in the virtual address space as configured by a combination of the stage1 and stage2 page tables, then the Cortex-A57 can issue a speculative instruction fetch to the first 64 bytes (cacheline) of the page with Execute-never (XN) permission. This speculative instruction fetch could result in a deadlock or introduce data integrity errors in the system by incorrectly accessing a read-sensitive device.

Note: This erratum corresponds to issue #3340 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

1) A Normal Write-Back Cacheable page precedes an Execute-never (XN) page in the virtual address space.
2) Instruction cache prefetch is enabled.

Implications

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

Workaround

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

2.7. Category C

808671: ETM might not output an Address packet immediately after a Trace Information packet.

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

The ETM generates periodic Synchronization packets due either to an external synchronization request or to the programming of the Synchronization Period Register, TRCSYNCPR. A Trace Information (Trace Info) packet is output as part of this periodic Synchronization packet. The ETMv4 Architecture specification recommends that an Address packet is output as soon as possible after a Trace Information packet.

Under certain conditions when ETM is tracing direct branches continuously, the ETM never sends the required Address packet after a Trace Information packet.

Note: This erratum matches bug #2861 in the ARM internal Jira database.

Configurations Affected

All

Conditions

The following sequence must occur.

1) The ETM is enabled.
2) Instruction tracing is currently active.
3) A synchronization packet is required, either because of an external synchronization request or because of the programming of TRSYNCPR.
4) The ETM outputs a Trace Information packet as part of the periodic Synchronization packet.
5) The ETM is tracing a continuous stream of direct branches.

In the above scenario, after step 4, the ETM never sends an address packet until either an indirect branch instruction is traced or the trace unit again transitions from the inactive state to the active state.

Implications

If trace is captured into an on-chip Embedded Trace Buffer (ETB) that overwrites old trace data with new trace data, then a trace decompressor might never be able to reconstruct the start address or the target address of the continuous direct branch P0 elements which are traced after the periodic Synchronization packet.

Workaround

There are 3 possible workarounds:

1) Tools must disable and re-enable the ETM at regular intervals when they expect large portions of trace to contain direct branch P0 elements.
2) Tools must program the ETM to disable and re-enable instruction trace at regular intervals when they expect large portions of trace to contain direct branch P0 elements. They can use one of the Counter resources as a timer for this disable and re-enable, for example by programming the Counter resource to decrement on every cycle, and reloading a value of either 100 or 200 whenever counter value becomes zero.
3) Tools must program the ETM to enable branch broadcast trace when they expect large portions of trace to contain direct branch P0 elements. However, branch broadcast trace can significantly increase the trace information output by the ETM. This additional trace bandwidth can cause trace buffer overflows and a resulting loss of trace information.

808871: Double bit ECC error on an invalid L1 data cache line can incorrectly trigger an imprecise abort

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

In the L1 data cache, if a single-bit ECC error occurs on the tag of an invalid cache line while the data array of the same cache line appears to have a double-bit ECC error, the core will incorrectly trigger an imprecise abort for a specific set of addresses.

Note: This erratum matches bug #2813 in the ARM internal Jira database.

Configurations Affected

All

Conditions

1) After system reset, the L1 data cache tag array is invalidated by hardware, and the L1 data cache data array has random data and ECC.
2) A single-bit ECC error in the L1 data cache tag array causes an invalid cache line to appear in either SharedClean or UniqClean state. The corresponding data and ECC of the same cache line in the L1 data array has a random value that appears to have a double-bit ECC error.
3) The physical address bits[44:14] of the data cache line are 0.
4) A load to the same cache line detects a single-bit ECC error on the tag and a double-bit ECC error on the data. Detecting both a double-bit ECC error and single-bit ECC error, the core will treat it as a double-bit ECC error, resulting in an imprecise abort. However, since this is an invalid line with a single-bit ECC error on the tag, imprecise abort should not occur.

Note that once a line is filled in the L1 data cache, the data and the ECC in the data array for that particular set/way is no longer susceptible to this erratum, even if the line is invalidated later. Therefore the window is quite small for this erratum to occur because it does not take long to populate the L1 data cache.

Implications

For a short period after system reset, a single-bit ECC error can be incorrectly classified as a double-bit ECC error causing an erroneous imprecise abort. This erratum is considered to be rare given it requires a load of a specific address range, a single-bit ECC error on the tag, a bogus double-bit ECC error on the data, and it must occur in a very small window after system reset.

The result is a very small increase in detected uncorrected error (DUE) failure in time (FIT) rate.

Workaround

There is no workaround.

809370: Watchpoint might be incorrectly taken on first half of unaligned ld/st crossing a 64-byte-aligned boundary

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

An unaligned memory transaction that crosses a 64-byte-aligned memory boundary is split into two requests in the core pipeline. Each of these requests is compared to the active watchpoints to see if a watchpoint event needs to occur.

If the address mask field in the DBGWCR is set to 'No mask at all' or 'No mask on VA[2:0] only', and the BAS field is set to match some bytes in the 1st or 2nd double word, then the first half of such an unaligned transaction will incorrectly trigger a watchpoint.

Note: This erratum matches bug #2870 in the ARM internal Jira database.

Configurations Affected

All

Conditions

1) Unaligned memory request crossing a 64-byte boundary, the first half to cacheline N (VA[48:6] = N) and the second to cacheline N+1 (VA[48:6] = N+1). DBGWVR has a watchpoint address in the lower 64-byte memory region (cacheline N) of the request.
2) DBGWCR[28:24] (Mask) = 5'b00000, 5'b00001 (reserved), 5'b00010 (reserved), 5'b00011, 5'b00100, or 5'b00101. Thus the active watchpoint includes the 1st or the 2nd double word of the cacheline N.
3) BAS field of DBGWCR is set to match some bytes in the 1st or 2nd double word.

Implications

An erroneous watchpoint will be taken on an incorrect but nearby address.

Workaround

If double word or finer granularity is not required, set the DBGWCR[28:24] to 5'b00110 (6-bits masked) or higher. This will result in a loss of 32-byte or finer granularity for a watchpoint.

Alternatively, software can detect false triggers on the watchpoint by looking for the following conditions:

1) The target watchpoint is to the lower 32 bytes of a cacheline but not the upper 32 bytes of the cacheline. This can be computed from DBGWVR and DBWCR.
2) When the watchpoint is triggered, the address reported in the DFAR register is in the upper 32 bytes of a cacheline.

811619: Return stack erroneous match for not taken indirect branch

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

When checking for a return stack match, the core fails to consider the branch taken bit. For an indirect branch not taken, there is no need to generate any address packet, and nothing should be changed on the return stack. However, because current RTL does not check for the branch taken bit, a possible match might be indicated for the not taken indirect branch. If this occurs, the top entry of the return stack will be popped due to the match. If the next taken indirect branch matches the second entry on the return stack, then no address will be generated. This will corrupt ETM decode. Note that if the next taken indirect branch matches the wrongly popped entry, an extra address packet will be generated, instead of only an atom packet. This additional address packet does not corrupt the ETM decoder and only impacts the trace bandwidth.

Note: This erratum matches bug #2964 in the ARM internal Jira database.

Configurations Affected

All

Conditions

The following sequence might fail;

1) Indirect branch (link) with address packet and return stack push
2) Indirect branch (link) with address packet and return stack push
3) Indirect branch (not link) not taken but target PC matches the top entry on the return stack (branch for step 2)
4) Indirect branch (not link) taken and the target PC matches the next entry on the return stack(branch for step 1)

For the indirect branch not taken (Step 3), there should not be an address packet generated anyway, and the return stack should not be affected. It should not be pushed or popped. However, the core will incorrectly indicate a return stack match for the not taken indirect branch, and will pop off the top entry (Step 2) from the stack. When the next taken indirect branch (Step 4) is executed, the target PC is compared with the second entry on the return stack (Step 1). An incorrect indication of match will drop the address packet and corrupt the decompressor.

Implications

The decoder cannot proceed because the address packet is missing from the trace stream. When the next sync point occurs, the decompressor can recover.

Occurrence of the failure sequence is limited to only SMC (self-modifying code). The indirect branches that cause the push onto the return stack have to be link, while the incorrectly matched not taken indirect branch has to be not link. Therefore a regular recursive code segment will not be able to trigger the error condition.

Workaround

The return stack has to be disabled. When disabling the return stack, an additional address packet will be generated. Hence the trace bandwidth will increase.

Other workarounds such as more synchronization might not completely fix the problem.

811672: When a single-bit ECC error occurs in the L2, uncorrected data might be returned

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

In the case of an unusual timing boundary condition, a single-bit ECC error in the L2 data array might not be corrected properly.

Note: This erratum matches bug #2916 in the ARM internal Jira database.

Configurations Affected

All systems where ECC is enabled and configured.

Conditions

1) A cache line fill to address A (CLF_A) hits the L2 and returns data to the L1 data cache.
2) The first 16-byte beat of data returning for CLF_A has a single-bit ECC error in the data.
3) Write streaming is enabled (CPUACTLR[28:25] != b1111) in the L1.
4) A full cache line-streaming write to address B (WR_B) is ready to issue to the L2. WR_B begins to issue during the same cycle the CLF_A incorrect data returns.
5) A store is in flight (ST_C) to the cache line of address A, or to the cache line being replaced by CLF_A.

Implications

Data corruption might occur if the erratum conditions exist. A single-bit error in the L2 data array might lead to incorrect data in the L1 data cache. A very small percentage of L2 data array single-bit ECC errors will be affected, because the error must be in the critical 16 bytes, must hit a narrow timing window, and must occur when store streaming is pushing full cache line writes to the L2 at the same time as a non-streaming store is hitting the cache line fill or the same set/way.

The result is a very small increase in silent data corruption (SDC) failure in time (FIT) rate in the presence of L2 data array errors.

Workaround

There is no workaround.

812170: Erroneous LR_mon on monitor trap of T32 ACTLR-disabled write accesses

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

The ACTLR (an IMPLEMENTATION DEFINED register) contains bits that control write accesses to other IMPLEMENTATION DEFINED registers (L2ACTLR, L2ECTLR, L2CTLR, CPUECTLR, CPUACTLR). When (1) EL3 is in AArch32 register width state, (2) an attempted T32 write (via MCR or MCRR) to any of L2ACTLR, L2ECTLR, L2CTLR, CPUECTLR, CPUACTLR occurs, (3) where the corresponding ACTLR control bit for the register attempting to be written is configured to disable the access, a monitor trap exception should and does occur. As part of the monitor trap exception entry sequence, the link register LR_mon should be updated with PC - 2 (when the trapped instruction was executed in Thumb state). Instead, LR_mon is updated with PC when the trapped instruction was executed in Thumb state. (LR_mon is correctly updated with PC - 4 when the trapped instruction was executed in ARM state).

Note: This erratum matches bug #2931 in the ARM internal Jira database.

Configurations Affected

Requires EL3 to be in AArch32 register width state. Requires ACTLR bits (any of L2ACTLR(ACTLR[6]), L2ECTLR(ACTLR[5]), L2CTLR(ACTLR[4]), CPUECTLR(ACTLR[1]), CPUACTLR(ACTLR[0])) to be cleared such that accesses to these registers is intentionally prevented. (Note that the reset value of these bits is indeed cleared, that is, accesses are disabled out of reset). Requires T32 write of L2ACTLR, L2ECTLR, L2CTLR, CPUECTLR, CPUACTLR.

Conditions

1) EL3 is using AArch32.
2) ACTLR bits (any of L2ACTLR(ACTLR[6]), L2ECTLR(ACTLR[5]), L2CTLR(ACTLR[4]), CPUECTLR(ACTLR[1]), CPUACTLR(ACTLR[0])) are cleared such that accesses to these registers are intentionally prevented. (Note that the reset value of these bits is indeed cleared, that is, accesses are disabled out of reset).
3) T32 write of L2ACTLR, L2ECTLR, L2CTLR, CPUECTLR, CPUACTLR using MCR or MCRR.

Implications

If the conditions above are met, LR_mon is PC instead of PC - 2.

Workaround

When handling monitor traps, software should perform the following steps:

1) Read SPSR[5] to check if the T bit is set
2) If so, read LR_mon and subtract 4 to obtain address X
3) Decode the instruction at address X
4) If the instruction at address X is MCR or MCRR, check if the targeted register is L2ACTLR, L2ECTLR, L2CTLR, CPUECTLR, or CPUACTLR
5) If the target is one of these registers, read the appropriate bit in the ACTLR to determine if access is prevented (L2ACTLR(ACTLR[6]), L2ECTLR(ACTLR[5]), L2CTLR(ACTLR[4]), CPUECTLR(ACTLR[1]), or CPUACTLR(ACTLR[0]))

If these conditions are all true, then the address in LR_mon is incorrect and needs to be decremented by 2.

812171: Erroneous ESR ISV for ARMv8-added A32/T32 load/store acquire/release exclusive instructions

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

The load/store exclusive instructions with acquire/release semantics added to A32 & T32 ISAs by the ARMv8 architecture, when exceptional and targeting EL2, erroneously set ESR_EL2 (AArch64) / HSR (AArch32) ISV to 1 (Instruction Syndrome Valid, that is, syndrome register bit[24]) . The architecture states that the ISV bit is 1 only for a restricted set of A32/T32 load and store instructions, where that set does not include any of the Exclusive forms of loads and stores. As a consequence of ISV being 1, other fields in the syndrome register (SAS (Syndrome Access Size), SSE (Syndrome Sign Extend), SRT (Syndrome Register Transfer), SF (Sixty-Four), AR (acquire/release semantics)) are not forced to a RES0 value but instead reflect the pertinent values for the exceptional instruction.

Note: This erratum matches bug #2978 in the ARM internal Jira database.

Configurations Affected

Requires A32 or T32 execution of any of {STLEXB, STLEXH, STLEX, STLEXD (Store Release Exclusive Byte/Halfword/Word/Dual), LDAEXB, LDAEXH, LDAEX, LDAEXD (Load Acquire Exclusive Byte/Halfword/Word/Dual), where the execution is exceptional and targets EL2. It is immaterial whether the register width state of EL2 is AArch32 or AArch64.

Conditions

Exceptional execution of A32 or T32 Store Release Exclusive Byte/Halfword/Word/Dual instructions or Load Acquire Exclusive Byte/Halfword/Word/Dual instructions where the exception target state is EL2.

Implications

If the conditions above are met, ESR ISV (Instruction Syndrome Valid) is 1 when it should be 0

Workaround

There is no workaround.

814669: L2 ECC error might cause a device memory type load to stall until the next timer interrupt

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

When the instruction flow has a mixture of loads of normal WB memory type and loads of device memory type, an L2 ECC error from the loads of normal WB memory type might prevent the loads of device memory type from making forward progress until the next interrupt.

Note: This erratum matches bug #3001 in the ARM internal Jira database.

Configurations Affected

All

Conditions

1) The instruction flow has a mixture of loads of normal WB memory type and loads of device memory type.
2) A load of device memory type crosses a 64-byte boundary. The high half of the device load allocates a fill buffer before the low half of the load allocates a fill buffer (this is not common but is possible).
3) A L2 ECC error occurs on a load of normal WB memory at the same cycle when the low half of the device load allocates a fill buffer.

Implications

If the above conditions occur, the CPU will stall until the next interrupt.

This erratum is considered rare given the requirement for a single bit ECC error colliding with an unaligned 64-byte crossing load and very unusual timing. Device accesses across 64-byte boundaries are uncommon. Therefore, this is not expected to be a serious issue for most systems.

Workaround

There is no workaround.

814670: A57 DMB barrier does not guarantee observation of the effects of a cache maintenance operation

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0

Description

A DMB following a cache maintenance operation that causes a copyback (WriteBack, WriteEvict, Evict or WriteNoSnoop transaction) to the system might complete before the copyback completes. This allows a memory operation to be performed after the barrier, possibly violating the architectural guarantees of the barrier. Only cache maintenance operations that cause a copyback, but do not require a CleanShared or CleanInvalid broadcast to the system, are affected. This includes all DCCSW, DCCISW and DCISW instructions, together with DCIMVA, DCCIMVA, DCCMVAU, and DCCMVAC instructions operating on non-shareable memory locations.

Note: This erratum matches bug #3004 in the ARM internal Jira database.

Configurations Affected

All

Conditions

1) DCCSW, DCCISW, DCISW, DCIMVA, DCCIMVA, DCCMVAU, or DCCMVAC to non-shareable memory targeting a cache line currently valid in the L2 cache.
2) DMB barrier.
3) Memory operation dependent on the order guarantees of the DMB.

Implications

In a coherent system with BROADCASTCACHEMAINT = 1'b1 and CMO set/way operations are only used for powerdown, this erratum will have no impact. Otherwise, if the above conditions occur, data corruption might occur caused by the memory operation following the DMB executing out-of-order with respect to the barrier. Note that the use of these instructions with non-shareable memory addresses is uncommon.

Workaround

Set CPUACTLR[58] to 1'b1 to disable DMB nullification. Note: this workaround might reduce system performance.

817170: Continuous streaming stores to device memory might block older stores

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1

Description

If continuous stores to nGnRnE/nGnRE/nGRE device memory are performed after a particular address pattern of stores to NC/WT/GRE memory, the core might prevent the older NC/WT/GRE stores from being written into memory.

Note: This erratum matches bug #3085 in the ARM internal Jira database.

Configurations Affected

All configurations

Conditions

1) A sequential stream of NC/WT/GRE stores of more than 4 cachelines is performed, triggering the core to predict that subsequent stores are sequential streaming stores and enter streaming mode. After the core establishes streaming mode, it holds off on issuing stores and attempts to gather a full cacheline to generate a single write transaction to the interconnect.
2) While waiting for a full cacheline of NC/WT/GRE stores, a continuous stream of nGnRnE/nGnRE/nGRE stores are issued at the end of the streaming NC/WT/GRE stores. The continuous nGnRnE/nGnRE/nGRE stores prevent the older NC/WT/GRE stores from being written into memory due to arbitration of shared resources always choosing the younger stores.

Implications

NC/WT/GRE stores in a particular address pattern might not be visible to the memory system in finite time if these stores are followed by continuous stores to nGnRnE/nGnRE/nGRE device memory. Note that the next interrupt will stop stores to nGnRnE/nGnRE/nGRE device memory, allowing the older NC/WT/GRE stores to be pushed out to the memory system.

Workaround

Insert a DMB between stores to NC/WT/GRE memory and stores to nGnRnE/nGnRE/nGRE device memory. Note that this workaround should only be pursued if the problem is observed. It is not believed that this erratum will surface in a real system.

817172: Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1

Description

If hardware detects a single-bit or double-bit ECC error on any RAM protected by ECC, the hardware initiates a sequence to correct the error. In the case of a single-bit ECC error, a read-modified-write sequence is generated to correct the single-bit error and write the corrected data back to the RAM. In the case of a double-bit error, the nINTERRIRQ signal is de-asserted signaling the presence of the double-bit error and the hardware writes the entry to resolve the error. If the double-bit error was associated with the L2 tag RAM or the L2 snoop tag RAM, the entry is invalidated. If the error was associated with the L2 dirty RAM then the entry is marked clean.

This erratum describes a condition when, after a single-bit ECC error detection, a double-bit ECC error is detected while the hardware performs the read-modified-write correction sequence. If this condition occurs, then nINTERRIRQ is not de-asserted to report the double-bit ECC error. The hardware correction sequence then proceeds to write the RAM to correct the error and might cause data corruption. Depending on the RAM associated with the error, corruption could occur to either the data, tag address, inclusion indicator or dirty status. This corrupted data or state could then be used by a subsequent transaction which might cause unpredictable results.

Note: This erratum matches bug #3070 in the ARM internal Jira database.

Configurations Affected

All configurations

Conditions

1) A single-bit ECC error detected on L2 RAM.

2) A double-bit ECC error detected on the same entry during hardware read-modified-write correction sequence.

Implications

If the above conditions occur, the RAM contents will be corrupted as the syndrome bits will be recalculated using the corrupted data and RAM updated with this corrupted data.

It is unpredictable as to the behavior of the system following this data corruption as it affects all of the RAMs in the L2 memory system which are protected by ECC.

If a double-bit error was detected initially there is no issue, because the hardware properly handles this case. If a single-bit error was detected initially and no double-bit error detected during the correction sequence there is no issue. The correction sequence duration is configuration controlled, but completes in less than 100 cycles, so this condition of a double-bit error being generated following a single-bit error detection is considered rare. For the majority of applications this is a minor error. In some applications the error might be more significant.

Workaround

There is no workaround.

817722: Read after read memory ordering requirements might be violated with specific sequences of exclusive stores and loads

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1

Description

Shortly after a STREX fails, if there is a specific sequence of store/load/load instructions targeting the same bytes of a cacheline that were targeted by the STREX, the processor might execute the two loads out of order and violate read after read ordering rules.

Note: This erratum matches bug #3101 in the ARM internal Jira database.

Configurations Affected

All configurations are affected.

Conditions

1) There is a code sequence of STREX/ST/LD/LD to the same cacheline. ST contains all bytes which are needed by the two LDs.
2) STREX hits a SharedClean line in the L1 Data Cache and therefore sends out a ReadUniq with Exclusive request, which turns into a CleanUniq with Exclusive request to the interconnect.
3) While the miss request of the STREX is outstanding, the ShareClean line is invalidated by a snoop.
4) The second LD gets forwarded data from the ST while the first LD cannot complete because of an unknown address or other rare microarchitectural hazards.
5) The STREX miss request comes back with a response of "exclusive monitor fail" and an "invalidate line" is returned. This can happen when two or more processors are contending for writing into the same memory location.
6) The ST sends out the miss request and merges data into the cache when the response come back. Next the cacheline is written back to L2, as a result of either normal capacity eviction or response to a CleanInvalidate type snoop.
7) Another processor performs a write affecting the same bytes the ST was modifying.
8) The first LD now completes and gets data, but it is the data that was modified by another processor.

Implications

If the above conditions are met, the processor might execute two loads out of order and violate read after read ordering rules.

Note that there is no functional concern for the normal spin lock using LDREX/STREX. When the STREX fails, the code may either loop back to the start of the spin lock or switch to code targeting different cachelines. In typical systems there should be no committed stores to the same cacheline as the lock being targeted by the STREX when the STREX fails.

Workaround

There is no workaround.

821570: Read following a write of a Timer TVAL register might return incorrect value

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

In Cortex-A57, the computation and update to the Timer CompareValue is done in one cycle, and the update to the read TimerValue is done in the next cycle. A write to a TVAL register updates the CompareValue. If a TVAL register is written and immediately followed by a read of the same TVAL register, there is insufficient time for the TimerValue to be updated and the wrong TimerValue is returned.

Note: This erratum matches bug #3122 in the ARM internal Jira database.

Configurations Affected

All configurations are affected.

Conditions

1) A write is performed to a Timer TVAL register.
2) A read is performed to the same Timer TVAL register in the very next cycle.

Implications

If the above conditions are met, the read will return the incorrect (old) TimerValue.

Workaround

Insert an ISB between the write and read. Only one cycle is needed between the two accesses to allow time for the TimerValue to be updated.

821571: Read or write of ICC_SRE_EL1(NS) might incorrectly generate an undefined exception

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0

Description

If the Cortex-A57 internal GIC CPU interface is enabled and configured in system register mode, a specific access might incorrectly generate an undefined exception when a hypervisor trap should have been taken.

Note: This erratum matches bug#3113 in the ARM internal Jira database.

Configurations Affected

Systems implementing EL1, EL2 and EL3 using AArch32 and the external pin GICCDISABLE=0 (that is, Internal GIC CPU interface is enabled).

Conditions

1) Operating in Symmetric System Register enabled mode, which has the system registers enabled for all exception levels.
2) ICC_SRE_EL[3,2].Enable=0.
3) In EL1NS, a read or write to the ICC_SRE_EL1(NS) register occurs.

Implications

When the above conditions occur, a hypervisor trap should be taken because of the ICC_SRE_EL2.Enable=0. However, an undefined exception is also generated because of the ICC_SRE_EL3.Enable=0. The undefined exception is taken because it has higher priority. As stated in the GICv3/GICv4 specification, Cortex-A57 should not have generated an undefined exception in this configuration.

Workaround

In this configuration, software should access the ICC_SRE_EL1(NS) register from EL2 exception level or higher.

822224: A57 might deadlock or have data corruption if the page attribute of a current page is remapped by another processor

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

When the OS/Hypervisor from one processor (PROC1) changes the page attribute of a current page from a device memory type to normal memory type or vice versa while the page is still currently used by another processor (PROC0), and PROC0 is executing a load or load multiple instruction that crosses a 64-byte cacheline boundary, PROC0 might deadlock.

Note: This erratum matches bug #3133 and bug #3142 in the ARM internal Jira database.

Configurations affected

Configurations with multiple processors in the system.

Conditions

1) A processor is executing a non-speculative load instruction whose memory access crosses a 64-byte cacheline boundary or a multiple register integer/neon load of device memory type.
2) The page being accessed by the load has been remapped from normal memory to device memory or from device memory to normal memory by another processor.

Implications

The processor might deadlock or have data corruption if the OS/Hypervisor from another processor changes the page attribute of a page currently being accessed non-speculatively by this processor from device memory to normal memory or vice versa. If the page is marked as invalid before the memory attribute changes, it greatly reduced the probability of deadlock.

Workaround

OS/hypervisor should avoid changing the page attribute from a device memory type to a normal memory type or vice versa while the page is accessed by a non-speculative load instruction from another processor.

822227: Using unsupported 16K translation granules might cause Cortex-A57 to incorrectly trigger a domain fault

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Cortex-A57 does not support 16K translation granules. If an AT instruction is executed targeting AArch64 EL1 and TCR_EL1 is programmed with a 16K translation granule, Cortex-A57 might incorrectly report a domain fault in PAR_EL1 register.

Note: This erratum matches bug #3139 in the ARM internal Jira database.

Configurations Affected

All configurations are affected.

Conditions

1) TCR_EL1[31] is programmed to zero (1'b0).
2) AT instruction is executed targeting AArch64 EL1.

Implications

If the above conditions are met, Cortex-A57 might incorrectly report a domain fault. Note that enabling 16K translation requires programming ECR_EL1[31] to 1'b0. This bit is RES1 in the Cortex-A57 TRM.

Workaround

There is no workaround.

822369: HSR.EC might be incorrect for certain accesses when HCR.TGE=1

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Cortex-A57 properly executes a HYP_TRAP instead of an UNDEFINED priority8 and lower priority exceptions in AArch32 from nonsecure user mode when HCR.TGE=1. However, the HSR.EC is returning 0x3 code when it should be returning the invalid/unknown 0x0 code (as described in section G1.10.6 "Routing general exceptions to EL2" of the ARM-ARMv8 specification). UNDEFINED priority7 exceptions correctly return 0x0.

Priority7 and Priority8 are as defined in section D1.13.2 "Synchronous exception prioritization" of ARM-ARMv8 specification.

Note: This erratum matches bug #3141 in the ARM internal Jira database.

Configurations affected

Systems with EL2 using AArch32 and HCR.TGE=1.

Conditions

1) EL2 is using AArch32.
2) HCR.TGE is programmed to 1.
3) An UNDEFINED (priority8 or lower priority) register access is executed from nonsecure user mode.
4) Hypervisor reads HSR after HYP_TRAP is executed.

Implications

Hypervisor will see the 0x3 HSR.EC value and see the exception is of the configurable trap, enable, disable class rather than the less explicit uncategorized reason class. In this case, it would have more information than the specification is indicating that it should have for this specific case.

Workaround

There is no workaround.

823969: Halting debug event occurring while ITSTATE is non-zero might cause subsequent instructions from the ITR to execute

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Description

Entering Debug state after execution of an IT instruction and before execution of the final instruction in an IT Block might result in erroneously applying ITSTATE conditions to the first few instructions executed from the EDITR while in Debug state. As such, these EDITR-sourced instructions are executed conditionally, whereas the correct behavior is to execute them unconditionally.

Note: This erratum matches bug #3158 in the ARM internal Jira database.

Configurations affected

Configurations where an Exception Level is configured as AArch32 register width state and another higher Exception Level is configured as AArch64 register width state.

Conditions

1) Debug state entered because of a Halting debug event, Breakpoint debug event, or Watchpoint debug event from AArch32 register width state, where any Exception Level above the current Exception Level is configured as AArch64 register width state.
2) Debug state entry occurs after executing an IT instruction and before executing the final instruction in the IT Block.

Implications

If the above conditions are met, instructions written into the EDITR are executed conditionally according to the condition that applies to subsequent instructions in the IT Block. For example, if Debug state entry occurs after execution of the first instruction in a 3-instruction IT Block and before execution of the second instruction in that 3-instruction IT Block, the first two instructions written into the EDITR are executed conditionally (as if they were the remaining two instructions in the IT Block), with the applied condition codes matching the condition codes that would have been applied to the remaining two instructions in the IT Block.

Workaround

Write four NOP instructions into the EDITR as the first instructions executed in debug state, to cause the ITSTATE bits to advance to 8'b0000_0000. Four NOP instructions will cause ITSTATE to transition to 8'b0000_0000 regardless of (a) the total number of instructions in the IT Block and (b) the entry point to Debug state relative to the number of remaining (unexecuted) instructions in the IT Block. After ITSTATE is zero, all subsequent EDITR-sourced instructions will be executed unconditionally.

825570: Access to Debug EDSCR.STATUS[5:2] register bits is Read-Write instead of Read-Only

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Entering Debug state updates EDSCR.STATUS[5:0] bits with the reason code for entering Debug state. As per the Debug architecture, access to EDSCR.STATUS[5:2] bits needs to be Read-only where as they are incorrectly implemented as Read-Write.

Note: This erratum matches bug #3184 in the ARM internal Jira database.

Configurations affected

All configurations are affected.

Conditions

1) Debug state entered because of a Halting debug event, Breakpoint debug event, or Watchpoint debug event.
2) External debugger writes a value which is different from the reason code for Debug state entry as captured in EDSCR.STATUS[5:2].

Implications

If the above conditions are met, until the processor exits Debug state, subsequent reads of EDSCR.STATUS[5:2] will return the last value written instead of the correct reason code for Debug state entry. Reads of EDSCR.STATUS[5:2] will correctly return zeroes when the processor is in Non-Debug state as required by Debug architecture.

Workaround

There is no workaround.

826975: Cortex-A57 might livelock because of exclusive load to non-shared memory location with mismatched attributes

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Cortex-A57 might livelock when executing a load exclusive to a memory location with one of the following memory attributes:

1) non-shared inner WT/outer WT
2) non-shared inner WB/outer WT
3) non-shared inner WB/outer NC
4) non-shared inner WT/outer WB
5) non-shared inner NC/outer WB

This erratum only occurs when the first 16 bytes of data returned and the last 16 bytes of data returned for the read request originating from the load exclusive are more than 512 cycles apart.

Note: This erratum matches bug #3189 in the ARM internal Jira database.

Configurations affected

All configurations are affected.

Conditions

1) There is a non-speculative load exclusive to a non-shared memory location with memory attributes as listed in the description above.
2) The load exclusive will send out a ReadNoSnoop request which asks for 64 bytes of data and thus needs 4 beats of 16-byte data transfer on the Cortex-A57 interface. The last 16-byte data beat arrives 512 cycles or more after the first 16-byte beat. This will cause the Cortex-A57 to release the fill buffer before satisfying the load exclusive.
3) The step 2) keeps repeating.

Implications

Cortex-A57 might livelock because of a load exclusive to a non-shared memory location with mismatched inner cacheability and outer cacheability and also to non-shared inner WT/outer WT memory locations when the return of the last beat of 16-byte data is always 512 cycles or more later than the return of the first beat of 16-byte data. Note that an interrupt will break the livelock.

Workaround

Avoid the use of load exclusive to non-shared memory locations with the above listed memory attributes.

826978: Cortex-A57 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Cortex-A57 might violate the read-after-read memory ordering requirement when a load forwards data from an older store which crosses a 16-byte boundary but not a 64-byte boundary. When a store instruction crosses a 16-byte boundary but not a 64-byte boundary, it is possible for the lower bytes to update the cache before the upper bytes update the cache. If the lower bytes update the L1 data cache, the line is then evicted from the L1 and modified by another master, and the line is brought back into the L1 data cache before the upper bytes are processed, there is a window where two loads that access the lower bytes of the store could violate the read-after-read memory ordering requirement. Specifically, an earlier load might see the data written by the other master, and the later load might see the data written by the store on the local processor.

Note: This erratum matches bug #3204 in the ARM internal Jira database.

Configurations affected

All configurations are affected.

Conditions

1) A store (ST1) is executed which crosses a 16-byte boundary but not a 64-byte boundary, modifying one or more bytes (A) below the 16-byte boundary and one or more bytes (B) above the 16-byte boundary.
2) After the lower bytes (A) of the store are written into cache, the cacheline is evicted or snooped out of L1D before the upper bytes of the store (B) update the cacheline.
3) The lower bytes (A) are modified by another memory master or processor (ST2).
4) The cache line is brought back into the L1 data cache (possibly by a load or preload instruction).
5) Two younger loads (LD1 and LD2) are executed that access the lower bytes modified by the store (A) before the upper bytes are written to the cache.

Implications

If the above conditions occur, it is possible that LD1 will return the modified data from the other master (ST2), but LD2 will see the data from the local store (ST1). This violates the following requirement in the ARM architecture specification (known as read-after-read ordering):

"It is impossible for an observer in the shareability domain of a memory location to observe two reads to the same memory location performed by the same observer in an order that would not occur in a sequential execution of a program."

The read after read ordering is significant in situations where the reads by one processor are racing in an unsynchronised manner with a write to the same location by a different processor. In these cases, it is expected that the algorithms involved will be relying on the write being seen as occurring in a single-copy atomic manner, such that the reads will see either the old or the new value, and not some amalgamation of the two. For this erratum to be observed, the write must span a 16-byte boundary, and so is not required architecturally to be single-copy atomic. Correspondingly it is very hard to envisage any situation where this erratum will be significant, and so it is characterised as being Cat C. Relatedly, this erratum has been present in volume shipping devices (Cortex-A15) without being observed in the field.

Workaround

No workaround is believed to be necessary for this erratum.

828023: Cortex-A57 might violate read-after-read memory ordering when the memory accesses of loads are not single-copy atomic

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1

Description

Cortex-A57 might violate the read-after-read memory ordering requirement when a load that is not single-copy atomic forwards data from an older store. Specifically, this might occur when a load instruction has more than two destination registers or requires a memory access of 16-byte or larger, such as LDM/VLDM with more than 2 elements in AArch32, LDP/LDNP with two Q registers or an Advanced SIMD load instruction with more than 2 elements in AArch64.

Such load instructions are split into multiple load micro operations which result in not being single-copy atomic. It is possible that the load instruction gets part of its data bytes from an older store and part of its data bytes from the cache or memory. For those data bytes obtained from the cache or memory, an earlier load might observe the newer modified data by another master and a later load might observe the older data before being modified by the other master.

Note: This erratum matches bug #3217 in the ARM internal Jira database.

Configurations affected

All configurations are affected.

Conditions

1) The code sequence requires 1 ST and 2 LD instructions.
   - ST1 [A + offset1]
   - LD2 [A + offset2]
   - LD3 [A + offset1]
- Where LD3 is a load instruction and is split into multiple load micro-ops inside CPU, say LD3.1 and LD3.2. ST1 is overlapped with part of LD3 (say LD3.1) and LD2 is overlapped with the other part of LD3 (say LD3.2).

2) The LD3 is executed out of order, before LD2. The first part of LD3 (LD3.1) gets its data from the ST1. The other part of LD3 instruction (LD3.2) gets it data from the cache. The MESI state of the cacheline [A] is SharedClean and there is an outstanding ReadUniq request to the cacheline [A].

3) Line [A] is snooped out of the L1 data cache and gets modified by another master. The new data is returned to the outstanding ReadUniq request.

4) LD2 instruction is executed now and observes the newer data than the 2nd part of LD3 instruction (LD3.2). This violates the read after read memory ordering as defined by the ARM architecture specification.

Implications

If the above conditions occur, it is possible that LD2 will observe the newer modified data from the other master, but part of LD3 will see the older data. This violates the following requirement in the ARM architecture specification (known as read-after-read ordering):

"It is impossible for an observer in the shareability domain of a memory location to observe two reads to the same memory location performed by the same observer in an order that would not occur in a sequential execution of a program."

The read after read ordering is significant in situations where the reads by one processor are racing in an unsynchronized manner with a write to the same location by a different processor. In these cases, it is expected that the algorithms involved will be relying on the reads being seen as occurring in a single-copy atomic manner. For this erratum to be observed, the load must not be a single-copy atomic load. Correspondingly it is very hard to envisage any situation where this erratum will be significant, and so it is characterized as being Category C.

Workaround

No workaround is necessary.

832076: AuxReg field of ID_MMFR0 reports incorrect value

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

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-A57 MPCore processor. However, because of this erratum, the AuxReg field in the ID_MMFR0 feature register indicates that these registers are not supported.

Note: This erratum matches bug #3248 in the ARM internal Jira database.

Configurations affected

All configurations 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 in Cortex-A57, 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.

832569: Cortex-A57 L1 prefetch works inefficiently for a load stream whose address is mapped to a page size of 64K or larger

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

Cortex-A57 L1 prefetcher loses its effectiveness for a load stream with a negative stride (a sequence of load instructions with decrementing addresses) and the address of the loads is mapped to a page size of 64KB or larger.

Note: This erratum matches bug #3249 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

For a load stream with a negative stride where the address of the loads is mapped to a page of 64KB or larger size, the prefetch address [15:12] is incorrect and always lags behind the demand loads.

Implications

Cortex-A57 L1 prefetch does not work efficiently for a load stream with negative stride and the address of the loads is mapped to a page of 64KB or larger size. Therefore, the L1 prefetcher does not help performance. Because the prefetch address is still within the same page of the demand load, it will not cause any functional issue. This erratum only affects performance.

Workaround

Set bit[47] of CPUACTLR_EL1 to make the L1 hardware prefetcher treat all page sizes of the loads as 4KB pages.

832769: HSTR.{T7,T8,T15} and HSTR_EL2.{T7,T8,T15} bits incorrectly trap CDP instructions

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

HSTR.Tx and HSTR_EL2.Tx bits are intended to trap MCR or MRC instructions with CRn set to cx and MCRR or MRRC instructions with CRm set to cx. For x = {7,8,15}, the trap bits do function in this capacity but also trap CDP/CDP2 instructions with CRn set to cx.

Note: This erratum matches bug #3262 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

1) a) EL2 is configured as register width state AArch32 and HSTR.Tx is set to 1.

or

1) b) EL2 is configured as register width state AArch64 and HSTR_EL2.Tx is set to 1.

2) A Non-secure CDP or CDP2 instruction with CRn set to X is executed in EL1 or EL0 using AArch32, where X is either 7, 8 or 15.

Implications

If condition 1a) and 2) are met, the CDP/CDP2 instruction will be erroneously trapped to Hyp mode. If condition 1b) and 2) are met, the CDP/CDP2 instruction will be erroneously trapped to EL2. In either case, the correct behavior is to take an UNDEF exception.

Workaround

There is no workaround.

833069: Disabling MMU Translation with CPUACTLR_EL1 "Enable Invalidates of BTB" bit set can cause Invalidate by PA or VA to fail

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

If the CPUACTLR_EL1 "Enable Invalidates of BTB" bit is set, an invalidate of the instruction cache by VA or PA operation executed in proximity to disabling of translation for the current EL (execution level) may not invalidate the proper instruction cache line. Note that this is not the default operating mode, and ARM recommends that the "Enable Invalidates of BTB" bit not be set.

Note: This erratum matches bug #3271 in the ARM internal JIRA database.

Configurations affected:

All configurations.

Conditions:

1) The "Enable Invalidates of BTB" bit (bit [0]) is set by a software write to the CPUACTLR_EL1 register.
2) The processor transitions from a level of MMU translation being enabled to being disabled. This is accomplished by two methods:
   - Transition from an execution level (EL) with the MMU enabled to an EL without the MMU enabled.
   - Software writes that change the state of the SCTLR_ELn.M bit for current execution level ELn.
3) At approximately the same time, an I-cache Invalidate by PA or I-cache Invalidate by VA operation is executed. This can originate from execution of an "IC IVAU" instruction upon any processor in the inner shareable domain or from an equivalent DVM command from a bus master.

Implications

Code that is thought to be no longer resident in the instruction cache may indeed still be resident. This can cause potential execution of stale code.

Workaround

The CPUACTLR_EL1 Enable Invalidates of BTB bit should not be set, so no workaround is required.

833472: PMU events 0x73 and 0x77 are not incremented for certain instructions

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

PMU events 0x73 (Operation speculatively executed - Data processing) and 0x77 (Operation speculatively executed - Crypto) are not incremented for certain instruction pairs.

Note: This erratum matches bug #3277 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

One or more of the following instruction pairs are executed:

1) A64 ADRP, ADD(immediate)
2) A64 MOVZ, MOVK
3) A32|T32 MOVW, MOVT
4) A32|T32|A64 AESE, AESMC
5) A32|T32|A64 AESD, AESIMC

Implications

In each of the instruction pairings above, under certain circumstances the appropriate PMU events (0x73 or 0x77) are not incremented correctly.

Workaround

No workaround is necessary.

834221: Trace packet might be corrupted in certain cases

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

If the AMBA Advanced Trace Bus (ATB) interface is run at the fastest ratio, which is divide-by-two of the main processor clock, it might be possible under certain frequency/voltage combinations for the Trace FIFO buffer pointers to become corrupted. The corruption is a result of incorrect multi-cycle path constraints that result in uncertainty at some clock edges in the ATB Trace FIFO buffer. This uncertainty typically occurs at inactive times due to the divide-by-two nature of the ATB interface. However, because of the coding style inside the ATB sync bridge, some registers are not clock gated at inactive times and incorrect or metastable values might be captured at the clock edge. If this occurs, the FIFO contents might be lost, resulting in a corrupted trace packet.

Note: This erratum matches bug #3295 in the ARM internal Jira database.

Configurations affected

Configurations with ATB running at divide-by-two of the main processor clock.

Conditions

Conditions for this erratum are implementation specific. There might be a specific combination of frequency, temperature and voltage that results in corrupted read/write pointers in the ATB Trace FIFO Buffers.

Implications

If the above conditions occur, the trace packet might be corrupted, resulting in a loss of trace data until the next synchronization point. Note that the ATB trace logic is designed to run at most divide-by-two of the processor clock, and this erratum is somewhat theoretical in nature. No failures have been observed in real silicon.

Workaround

- Configure the ATB bus to run slower than divide-by-two of the processor.

or

- When capturing traces, limit the processor frequency to whatever timing was achievable in a single cycle in the ATB syncbridge implementation. This would be half of the max processor frequency in the worst case.

834469: PMU event counters are incorrectly permitted to count events

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

SDER32_EL3.SUNIDEN is intended to enable PMU event counters while executing instructions in Secure EL0 mode when Secure EL3 or Secure EL1 is configured with register state width AArch32. Because of this erratum, SDER32_EL3.SUNIDEN might incorrectly enable PMU event counters while executing instructions in Secure EL0 mode when Secure EL1 is configured with register state width AArch64.

Note: This erratum matches bug #3300 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

1) Secure EL1 is configured as register width state AArch64.
2) MDCR_EL3.SPME is 0.
3) External debug authentication signals SPIDEN and SPNIDEN are LOW.
4) SDER32_EL3.SUNIDEN is 1.
5) Exceptional level is EL0 and secure state.
6) PMU event counters are enabled.

Implications

If the above conditions are met then PMU event counters might incorrectly increment when corresponding events occur.

Note: SDER32_EL3.SUNIDEN is reset to 0 in hardware. If the secure EL1 register width state never switches between AArch64 and AArch32, then this erratum scenario will not occur.

Workaround

If the secure EL1 register width state switches between AArch64 and AArch32, then the secure monitor can context switch SDER32_EL3 when moving to AArch64 state and force SDER32_EL3.SUNIDEN to 0.

834528: PMU event BUS_CYCLES might be incorrect in some cases

Category C
Products Affected: Cortex-A57 MPCore.
Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

PMU event 0x1D (BUS_CYCLES) might be incorrect in some scenarios.

Note: This erratum matches bug #3280 in the ARM internal JIRA database.

Configurations affected

All configurations affected.

Conditions

1) L2 clock is gated after 256 cycles of inactivity.
2) BUS_CYCLES event is read.

Implications

If the above conditions are met, the BUS_CYCLES count PMU event will be incorrect.

Workaround

If accurate BUS_CYCLES counts are required during periods of L2 inactivity, dynamic clock gating of the L2 logic can be disabled by setting L2ACTLR_EL1[27] = 1'b1. However, this is not recommended for normal operation because it will result in increased power consumption.

834920: Write of JMCR in EL0 does not generate an undefined exception

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2

Description

When EL0 is using AArch32 register width state and a write is performed in EL0 to JMCR, Cortex-A57 allows the write to occur. The intended behavior is to generate an undefined exception.

Note: This erratum matches bug#3304 in the ARM internal Jira database.

Configurations Affected

Systems with EL0 using AArch32 register width state.

Conditions

1) System in EL0 using AArch32.
2) Write is performed to JMCR.

Implications

An undefined exception should occur when the above conditions occur. Cortex-A57 incorrectly allows the write to go through, although no physical register exists for JMCR and therefore the write has no effect on processor state.

Workaround

There is no workaround.

844270: Syndrome value incorrect for software-induced Virtual Abort exception

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

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

Note: This erratum corresponds to issue #3314 in the ARM internal tracking system.

Configurations affected

Cortex-A57 systems with EL2 implemented and using AArch32.

Conditions

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

Implications

If the above conditions occur, a virtual asynchronous abort will be correctly taken, but the DFSR syndrome value will be all zeroes, which is incorrect. This erratum is not expected to impact any existing systems. It requires a 32-bit hypervisor using virtual asynchronous aborts to generate exceptions in guest operating systems. There is no known hypervisor for Cortex-A57 that meets these conditions. Also, even if utilized, a virtual asynchronous abort of this type would be a fatal error, making the syndrome not helpful in system recovery.

Workaround

No workaround is required.

850774: Cortex-A57 incorrectly allows access to GICv3 common registers in a specific configuration

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

If the Cortex-A57 internal GIC CPU interface is enabled and configured in system register mode, a specific access to one of the GICv3 common registers with ICH_HCR_EL2.TC=1 might incorrectly allow an access when an UNDEFINED exception or Monitor Trap should have been taken. ICH_HCR_EL2.TC=1 should not have any effect on GICv3 common register accesses in EL1S or EL2.

Note: This erratum corresponds to issue #3321 in the ARM internal tracking system.

Configurations Affected

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

Conditions

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

Implications

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

Workaround

There is no workaround.

850775: ATB stall from trace subsystem might deadlock the processor

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

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

Note: This erratum corresponds to issue #3322 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

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

Implications

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

Workaround

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

851023: Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

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

Note: This erratum corresponds to issue # 3323 the ARM internal tracking system.

Configurations affected

Coherent ACE systems that enable WriteUnique/WriteLineUnique transactions.

Conditions

1) WriteUnique/WriteLineUnique transactions are enabled by changing the default value of L2ACTLR[4] and setting it to 1’b0.
2) A Cortex-A57 processor issues a Write-Back No-Allocate store (OP1). This can be a streaming store that was downgraded to Write-Back No-Allocate by the processor.
3) There is a pending eviction, forcing (OP1) to stall because of ACE requirements that require outstanding evictions to complete before WriteUnique/WriteLineUnique stores are performed.
4) A continuous stream of L2 cache linefills occurs, from other cores and/or prefetch, which triggers new evictions.
5) There is significant sustained backpressure in the interconnect, which keeps the system backed up and the ACE write channel queue near full.
6) Specific arbitration and timing conditions exist which, when combined with condition 5), trigger a microarchitectural hazard that causes condition 3) to repeat.

Implications

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

Workaround

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

852579: Trace On packets from ETM are not generated in specific conditions around Debug Halt exceptions

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

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

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

Note: This erratum corresponds to issue #3327 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

Condition Set A

1) ETM is enabled.
2) The core halts and a Debug Halt exception is traced.
3) a) An exception occurs on the first instruction after leaving Debug state.

   or
4) b) A processor warm reset occurs while in Debug state

Or

Condition Set B

1) ETM is enabled.
2) Trace becomes active.
3) One or more instructions are executed, but no P0 elements.
4) The core halts with the resulting Debug Halt exception being traced.

Implications

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

Or

If Condition Set B occurs, then a Trace On packet is not output at step 2. The address and context are output correctly for the first instruction that is traced. The missing Trace On element means that a trace analyzer might not highlight the gap in the trace for the execution that was not traced before step 2.

Workaround

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

For both sets of conditions, a workaround is to avoid using ViewInst filtering address regions when using Trace with Halt exceptions.

852580: Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

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-A57 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-A57 does not force completion of the packet prior to acknowledging the flush request. This is a violation of the ETMv4 architecture.

Note: This erratum corresponds to issue #3328 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

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

Implications

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

Workaround

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

852581: Trace Context packet might not be output on a Reset or System Error exception

Category C
Products Affected: Cortex-A57 MPCore.
Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

In some scenarios, Cortex-A57 might not output a Context packet for the handler of the exception over the AMBA Trace Bus (ATB) when a reset exception or System Error exception causes an Exception level change. In all scenarios, the required Trace on, Exception, and Address trace packets are properly encoded and output.

Note: This erratum corresponds to issue #3329 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

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

Implications

If the above conditions occur, a Context packet is not output on AMBA ATB, as required by the architecture.

For the reset exception, software can assume that the Exception level will always change to EL3 coming out of reset.

A System Error might be routed to different Exception levels, so it would require software to query processor configuration registers bits to determine the Exception level for the context change.

Workaround

Enable CID and VMID tracing.

853971: An external data snoop might cause data corruption when an Evict transaction is pending

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3

Description

If Cortex-A57 is configured to send Evict transactions for a cache line in UniqueClean (UC) state, then Cortex-A57 might return stale data on a snoop response.

Note: This erratum corresponds to issue #3333 in the ARM internal tracking system.

Configurations affected

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

Conditions

1) Cortex-A57 is configured not to send data for UC evictions. This is accomplished by setting the value of L2ACTLR[14] to 1’b0. The default value of this bit is 1'b1.
2) Cortex-A57 is configured to push dataless Evict transactions to the external world. This is accomplished by setting the value of L2ACTLR[3] to 1’b0. The default value of this bit is 1'b1.
3) Cortex-A57 issues a dataless Evict transaction (one of SnpClean, SnpShared, and SnpUnique transactions) for a cache line in UC state.
4) Cortex-A57 issues an Evict transaction and is waiting for a completion response (COMP) from the interconnect.
5) The interconnect issues a COMP followed by a snoop request to the same cache line address.
6) Cortex-A57 receives the snoop request before the COMP because of a race condition in the interconnect.

Implications

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

Workaround

There are multiple workarounds:

1) Configure Cortex-A57 to send data for UC evictions. This can be accomplished by setting L2ACTLR[14] to 1’b1.

Or

2) Configure Cortex-A57 to not push Clean/Evict transactions to the external world. This can be accomplished by setting L2ACTLR[3] to 1’b1.

681545: Trace On packets from ETM are not generated in specific conditions around System Error exceptions

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3335 in the ARM internal tracking system.

Configurations affected

All configurations are affected.

Conditions

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

Implications

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

Workaround

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

1185470: Exception packet for return stack match might return incorrect [E1:E0] field

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

When an abort or trap is taken at the target of an indirect branch matching the return stack value in the core ETM, an Exception packet might be generated with the 2-bit field [E1:E0] = 0b10, which implies an Address element before the Exception element. When there is a trace return stack match, an Address element should not be generated before the Exception element. With [E1:E0] = 0b10, the external Trace Analyzer might read the trace packet sequence to expect an Address element output before the Exception element and not complete the stack pop, which is incorrect. The correct value in the [E1:E0] field in the Exception packet for this case, should be 0b01.

Note: This erratum corresponds to issue #3341 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) ETM is enabled.

2) TRCCONFIGR.RS = 1, which indicates the return stack is enabled.

3) Abort or trap is taken at the target of an indirect branch matching the return stack.

Implications

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

Workaround

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

1328360: Speculative TLB fills might occur past a DSB instruction

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3343 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

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

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

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

Implications

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

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

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

Workaround

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

1406397: TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3346 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

1) TCR_EL1.AS=0.

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

Implications

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

Workaround

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

1572968: Younger load incorrectly reporting a synchronous external abort due to an older load detecting an asynchronous external abort

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3347 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

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

Implications

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

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

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

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

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

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

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

with the higher privileged software, and it is expected that this will be considered fatal and the physical machine will need to be reset.

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

Workaround

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

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

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

1742085: Older load incorrectly reporting a synchronous external abort instead of a permission or domain fault due to a younger load detecting a synchronous external abort

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3349 in the ARM internal tracking system.

Configurations affected

This erratum affects all configurations.

Conditions

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

Implications

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

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

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

Workaround

There is no workaround.

1742089: Writes to normal memory might not be made globally observed in a finite amount of time when core is actively in write streaming mode

Category C

Products Affected: Cortex-A57 MPCore.

Present in: r0p0, r0p1, r1p0, r1p1, r1p2, r1p3. Open.

Description

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

Note: This erratum corresponds to issue #3350 in the ARM internal tracking system.

Configurations affected

This erratum affects all multi-core configurations.

Conditions

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

Implications

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

Workaround

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