ARM Processor
Cortex®-A7 MPCore

Product Revision r0p2, r0p3, r0p4, r0p5

Software Developers Errata Notice

Non-Confidential - Released

ARM®

Software Developers Errata Notice

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-016887
- 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.

06 May 2016: Changes in Document v14

Page	Status	ID	Cat	Rare	Summary of Erratum
24	New	856125	CatB		Stage 2 XN attribute is suppressed when stage 1 MMU is disabled

23 Feb 2015: Changes in Document v13

Page	Status	ID	Cat	Rare	Summary of Erratum
23	New	844169	CatB		Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set

06 Aug 2014: Changes in Document v12

Page	Status	ID	Cat	Rare	Summary of Erratum
12	Updated	823274	CatA	Rare	Load or store which fails condition code check might cause deadlock or data corruption
29	Updated	790569	CatC		Accesses to debug data transfer registers when the OS Lock is set behave incorrectly

28 Feb 2014: Changes in Document v11

Page	Status	ID	Cat	Rare	Summary of Erratum
12	New	823274	CatA	Rare	Load or store which fails condition code check might cause deadlock or data corruption

25 Nov 2013: Changes in Document v10

Page	Status	ID	Cat	Rare	Summary of Erratum
22	New	814220	CatB		Cache maintenance by set/way operations can execute out of order
43	New	809719	CatC		PMU events 0x07, 0x0C, and 0x0E do not increment correctly

11 Jul 2013: Changes in Document v9

Page	Status	ID	Cat	Rare	Summary of Erratum
19	Updated	802022	CatB		A CPU can interfere with the duplicate tag RAM invalidation process for another CPU and cause deadlock
41	New	804069	CatC		Exception mask bits are cleared when an exception is taken in Hyp mode
42	New	805420	CatC		PMU event counter 0x14 does not increment correctly

16 Apr 2013: Changes in Document v8

Page	Status	ID	Cat	Rare	Summary of Erratum
37	New	802119	CatC		Debug registers DBGLSR, DBGDEVID, and DBGDEVID1 are only accessible during powerdown if DBGSWENABLE[CPU] is asserted
39	New	803219	CatC		TBB/TBH incorrect branch address is not mis-predicted
40	New	803269	CatC		System instruction accessing CNTPCT or CPACR might be ignored in debug state

21 Mar 2013: Changes in Document v7

Page	Status	ID	Cat	Rare	Summary of Erratum
17	New	801872	CatB		Domain faults can be incorrectly reported in the PAR for ATS1C* operations executed in Hyp mode
19	New	802022	CatB		A CPU can interfere with the duplicate tag RAM invalidation process for another CPU and cause deadlock
35	New	799972	CatC		Non-cacheable request event incorrectly counts cacheable instruction accesses
36	New	800721	CatC		Reads of the DBGPCSR can cause incorrect values to be latched into DBGCIDSR and DBGVIDSR

07 Nov 2012: Changes in Document v6

No new or updated errata in this document version.

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

19 Oct 2012: Changes in Document v5

Page	Status	ID	Cat	Rare	Summary of Erratum
31	New	791620	CatC		A non-cacheable store in a tight loop might not become observable until the loop completes
32	New	792169	CatC		A read of DBGOSLSR during powerdown returns an incorrect value
33	New	793369	CatC		Loads and stores with mismatched attributes might cause deadlock
34	New	794322	CatC		An instruction fetch can be allocated into the L2 cache after the cache is disabled

18 Jul 2012: Changes in Document v4

Page	Status	ID	Cat	Rare	Summary of Erratum
16	New	789420	CatB		Hardware virtual interrupt deactivates incorrect physical interrupt
29	New	790569	CatC		Accesses to debug data transfer registers when the OS Lock is set behave incorrectly
30	New	790570	CatC		Accesses to debug registers when the OS Double Lock is set may result in incorrect behavior

25 May 2012: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
15	New	784478	CatB		CTIINTACK register needs clearing each time it is set
27	New	784472	CatC		CTI Authentication Status register is incorrect
28	New	784519	CatC		DBGPRCR.CORENPDRQ incorrectly changed on warm reset

26 Apr 2012: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
13	New	781670/783069	CatB		HSR Incorrect for Advanced SIMD / VFP instructions made UNDEFINED in Hyp mode by HCPTR settings
26	New	781169	CatC		DBGPRSR[1:0] are incorrectly RAZ during debug over powerdown

15 Mar 2012: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
25	New	777869	CatC		DBGDSCRext[15:14] is masked when DBGEN is LOW and OS Lock is SET

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-A7 MPCore processor.

1.1. Scope of this document

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

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

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

1.2. Categorization of errata

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

Table 1 Categorization of errata

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

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

The 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	r0p2	r0p3	r0p4	r0p5
823274	CatA	Rare	Load or store which fails condition code check might cause deadlock or data corruption	X	X	X	X
856125	CatB		Stage 2 XN attribute is suppressed when stage 1 MMU is disabled	X	X	X	X
844169	CatB		Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set	X	X	X	X
814220	CatB		Cache maintenance by set/way operations can execute out of order	X	X	X	X
802022	CatB		A CPU can interfere with the duplicate tag RAM invalidation process for another CPU and cause deadlock	X	X	X	
801872	CatB		Domain faults can be incorrectly reported in the PAR for ATS1C* operations executed in Hyp mode	X	X	X	
789420	CatB		Hardware virtual interrupt deactivates incorrect physical interrupt	X	X		
784478	CatB		CTIINTACK register needs clearing each time it is set	X			
783069	CatB		HSR Incorrect for Advanced SIMD / VFP instructions made UNDEFINED in Hyp mode by HCPTR settings	X			
781670	CatB		HSR Incorrect for Advanced SIMD / VFP instructions made UNDEFINED in Hyp mode by HCPTR settings	X			
809719	CatC		PMU events 0x07, 0x0C, and 0x0E do not increment correctly	X	X	X	X
805420	CatC		PMU event counter 0x14 does not increment correctly	X	X	X	X
804069	CatC		Exception mask bits are cleared when an exception is taken in Hyp mode	X	X	X	X
803269	CatC		System instruction accessing CNTPCT or CPACR might be ignored in debug state	X	X	X	
803219	CatC		TBB/TBH incorrect branch address is not mis-predicted	X	X	X	

ID	Cat	Rare	Summary of Erratum	r0p2	r0p3	r0p4	r0p5
802119	CatC		Debug registers DBGLSR, DBGDEVID, and DBGDEVID1 are only accessible during powerdown if DBGSWENABLE[CPU] is asserted	X	X	X	
800721	CatC		Reads of the DBGPCSR can cause incorrect values to be latched into DBGCIDSR and DBGVIDSR	X	X	X	
799972	CatC		Non-cacheable request event incorrectly counts cacheable instruction accesses	X	X	X	
794322	CatC		An instruction fetch can be allocated into the L2 cache after the cache is disabled	X	X		
793369	CatC		Loads and stores with mismatched attributes might cause deadlock	X	X		
792169	CatC		A read of DBGOSLSR during powerdown returns an incorrect value	X	X		
791620	CatC		A non-cacheable store in a tight loop might not become observable until the loop completes	X	X		
790570	CatC		Accesses to debug registers when the OS Double Lock is set may result in incorrect behavior	X	X		
790569	CatC		Accesses to debug data transfer registers when the OS Lock is set behave incorrectly	X	X		
784519	CatC		DBGPRCR.CORENPDRQ incorrectly changed on warm reset	X			
784472	CatC		CTI Authentication Status register is incorrect	X			
781169	CatC		DBGPRSR[1:0] are incorrectly RAZ during debug over powerdown	X			
777869	CatC		DBGDSCRext[15:14] is masked when DBGEN is LOW and OS Lock is SET	X			

2.2.1. r0p5 implementation fix

Note the following erratum might be fixed in some implementations of r0p5. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[0]	823274	Programmer CatA	Rare	A Load or store which fails condition code check might cause deadlock or data corruption

Note that there is no change to the MIDR which remains at r0p5 but the REVIDR is updated from 0x00 to 0x01 to indicate that one erratum is corrected. Software will identify this release through the combination of MIDR and REVIDR.

2.2.2. r0p4 implementation fix

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

REVIDR[0]	802022	Programmer CatB		A CPU can interfere with the duplicate tag RAM invalidation process for another CPU and cause deadlock

Note that there is no change to the MIDR which remains at r0p4 but the REVIDR is updated from 0x00 to 0x01 to indicate that one erratum is corrected. Software will identify this release through the combination of MIDR and REVIDR.

2.2.3. r0p3 implementation fixes

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

REVIDR[0]	802022	Programmer CatB		A CPU can interfere with the duplicate tag RAM invalidation process for another CPU and cause deadlock
REVIDR[1]	823274	Programmer CatA	Rare	A Load or store which fails condition code check might cause deadlock or data corruption

Note that there is no change to the MIDR which remains at r0p3 but the REVIDR is updated from 0x00 to 0x03 to indicate that two errata are corrected. Software will identify this release through the combination of MIDR and REVIDR.

2.3. Category A

There are no errata in this category.

2.4. Category A (Rare)

823274: Load or store which fails condition code check might cause deadlock or data corruption

Category A Rare

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

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

Configurations Affected

This erratum does not affect the processor if one of the following is true:

- The variant and revision reported by the MIDR is r0p3 and REVIDR[1] is set to 1.
- The variant and revision reported by the MIDR is r0p5 and REVIDR[0] is set to 1.

To be affected by this erratum, the Cortex-A7 processor must be configured with VFP or Neon support.

Conditions

1) A VFP divide or square root operation is executed.
2) Within the subsequent 58 instructions, a load or store is executed which causes a pipeline stall (caused by a cache miss or memory system contention).
3) Between the VFP divide or square root and the load/store there are none of the following instructions:
   - WFI/WFE
   - An instruction that reads the result of the divide or square root.
   - An instruction that reads the FPSCR.
4) Within the subsequent six instructions a load instruction is executed that satisfies the following requirements:
   - The instruction is conditional.
   - The instruction fails its condition code check.
   - The data addressed by the instruction given correct register inputs crosses an 8-byte boundary (including cases where an alignment fault would have been generated if the instruction had passed its condition code check).
5) Between the earlier load/store instruction and the later load instruction is an instruction which writes to a register used in the address calculation of the load.
   - This instruction must have the opposite condition code to the load instruction.
   - There must be no instructions which set the condition code flags between this instruction and the later load instruction.
6) While the pipeline is stalled due to the earlier load or store, the VFP divide or square root completes.

If these conditions are met, then under rare conditions data memory might be corrupted, which might lead to a deadlock.

Implications

It is believed that the various conditions required to cause this erratum are very unlikely to occur together in practice. This erratum has never been observed in deployed products. As such, the practical implications for real systems should be minimal.

Workaround

There is no workaround for this erratum.

2.5. Category B

781670: HSR Incorrect for Advanced SIMD / VFP instructions made UNDEFINED in Hyp mode by HCPTR settings

Category B

Products Affected: Cortex-A7 Floating Point Unit.

Present in: r0p2

Description

The Cortex-A7 MPCore Processor implements the ARMv7 Virtualization Extensions and can be implemented to include Advanced SIMD v2 and VFP v4. An Advanced SIMD or VFP instruction made UNDEFINED in Hyp mode by the programming of the HCPTR should write 0x7 to the HSR EC field. Because of this erratum, the Undefined Instruction exception writes 0x0 to the HSR EC field instead.

Configurations Affected

To be affected by this erratum, the processor implementation must include VFP v4 or Advanced SIMD v2 and VFP v4.

Conditions

- Use of Advanced SIMD and VFP instructions is enabled in the FPEXC register.
- The HCPTR is programmed to trap Non-secure use of Advanced SIMD and/or VFP instructions to Hyp mode.
- An Advanced SIMD or VFP instruction is executed in Hyp mode.

Implications

If the HCPTR is used to trap Non-secure use of Advanced SIMD or VFP instructions, the PL2 Undefined Instruction exception handler can not use the value of the HSR EC field to determine if an exception is due to an UNKNOWN cause or an Advanced SIMD or VFP instruction made UNDEFINED because of the programming of the HCPTR.

Workaround

If an Advanced SIMD or VFP instruction may be UNDEFINED in Hyp mode because of the programming of the HCPTR, the PL2 Undefined Instruction exception handler can not use the value of the HSR EC field to determine whether to check the programming of the HCPTR. The Undefined Instruction exception handler must instead load the UNDEFINED instruction from memory and examine it to determine whether it is an Advanced SIMD or VFP instruction.

783069: HSR Incorrect for Advanced SIMD / VFP instructions made UNDEFINED in Hyp mode by HCPTR settings

Category B

Products Affected: Cortex-A7 NEON Media Engine.

Present in: r0p2

Description

The Cortex-A7 MPCore Processor implements the ARMv7 Virtualization Extensions and can be implemented to include Advanced SIMD v2 and VFP v4. An Advanced SIMD or VFP instruction made UNDEFINED in Hyp mode by the programming of the HCPTR should write 0x7 to the HSR EC field. Because of this erratum, the Undefined Instruction exception writes 0x0 to the HSR EC field instead.

Configurations Affected

To be affected by this erratum, the processor implementation must include VFP v4 or Advanced SIMD v2 and VFP v4.

Conditions

- Use of Advanced SIMD and VFP instructions is enabled in the FPEXC register.
- The HCPTR is programmed to trap Non-secure use of Advanced SIMD and/or VFP instructions to Hyp mode.
- An Advanced SIMD or VFP instruction is executed in Hyp mode.

Implications

If the HCPTR is used to trap Non-secure use of Advanced SIMD or VFP instructions, the PL2 Undefined Instruction exception handler can not use the value of the HSR EC field to determine if an exception is due to an UNKNOWN cause or an Advanced SIMD or VFP instruction made UNDEFINED because of the programming of the HCPTR.

Workaround

If an Advanced SIMD or VFP instruction may be UNDEFINED in Hyp mode because of the programming of the HCPTR, the PL2 Undefined Instruction exception handler can not use the value of the HSR EC field to determine whether to check the programming of the HCPTR. The Undefined Instruction exception handler must instead load the UNDEFINED instruction from memory and examine it to determine whether it is an Advanced SIMD or VFP instruction.

784478: CTIINTACK register needs clearing each time it is set

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2

Description

The CTI includes a CTIINTACK register that software can use to acknowledge a trigger instead of using the hardware acknowledge **CTITRIGOUTACK** signal. The correct operation of this register is that writing a one to the bit corresponding to a trigger output causes that trigger to be cleared, and this does not affect future triggers.

Because of this erratum, when a bit in the CTIINTACK register is set to one, it remains set until cleared by writing zero to the register. This causes the corresponding trigger output to be acknowledged immediately if it occurs again, which can lead to the trigger being missed.

The CTIINTACK register is normally used in two cases:

- To clear a debug related interrupt, if required by the interrupt controller.
- To clear a debug entry request generated by another processor, when using cross-halting.

Configurations Affected

This erratum affects implementations of the processor that include the integration layer.

Conditions

The following sequence of conditions must occur:

1) A CTI trigger output fires.
2) The software writes a one to the corresponding bit of the CTI CTIINTACK register to acknowledge the trigger output.
3) The same trigger output fires again before the corresponding bit in the CTIINTACK register is cleared to zero.

Implications

Trigger outputs might be missed:

- In the case of a debug related interrupt that uses CTIINTACK to clear the interrupt, for events other than the first event.
- In the case of a cross-halting debug request, after the first time a processor halts and restarts, a subsequent halt might not halt any other processors.

Workaround

This is a workaround for tools vendors.

When the CTIINTACK register is written with a nonzero value, it must be immediately written to again with the value zero. This prevents any future events on the corresponding trigger output from being acknowledged.

If this workaround is used, there remains a race condition, whereby a trigger output that occurs between the two register writes might be lost. This is unlikely to be significant, because the timing of trigger outputs and the timing of register writes are not highly correlated, and if the trigger output had occurred before the first register write, then it would also have been lost.

789420: Hardware virtual interrupt deactivates incorrect physical interrupt

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

The Cortex-A7 MPCore processor can contain an integrated ARM Generic Interrupt Controller (GIC) that supports deactivating the originating physical interrupt when software deactivates a virtual interrupt. Under some circumstances the wrong physical interrupt might be deactivated.

Configurations Affected

This erratum only affects processors implemented with the GIC.

Conditions

Two or more List Registers (GICH_LRn) contain the same VirtualID, of which:

- One is an active hardware virtual interrupt, i.e. State==10 and HW==1, and
- At least one has a non-zero PhysicalID that is different to the PhysicalID of the active interrupt.

If these conditions are met then deactivating the active virtual interrupt might result in the wrong physical interrupt being deactivated. Virtual interrupts can be deactivated using the GICV_EOIR, GICV_AEOIR or GICV_DIR, depending on the value of GICC_CTLR.EOImode and the group of the virtual interrupt.

Implications

If this erratum occurs:

- the incorrectly deactivated physical interrupt might be signaled to the CPU again
- the physical interrupt that should have been deactivated, but was not, might never again be signaled to the CPU.

Workaround

Software can avoid this erratum by ensuring that when a List Register is programmed with a new virtual interrupt none of the other List Registers hold the same VirtualID field.

Two possible methods of achieving this are:

1) When programming a new virtual interrupt, reusing the last List Register that contained the required VirtualID, if it has not since been overwritten with a different VirtualID.
2) Writing all invalid List Registers to 0x0 before programming a new virtual interrupt.

801872: Domain faults can be incorrectly reported in the PAR for ATS1C* operations executed in Hyp mode

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The Cortex-A7 MPCore processor implements the ARM Virtualization Extensions, and therefore implements Hyp mode. When executing in Hyp mode, software can use the ATS1C* operations to perform address translation operations in the context of the Non-secure PL1 modes. The result of an operation is written to the PAR. This erratum means that in some cases the information written to the PAR might be incorrect. ATS12NSO* operations are not affected by this erratum.

Configurations Affected

All configurations.

Conditions

Either:

1) The processor is executing in Hyp mode.
2) HSCTLR.M has the value 0, disabling the PL2 stage 1 MMU.
3) The Non-secure copy of SCTLR.M has the value 1, enabling the PL1&0 stage 1 MMU.
4) The Non-secure copy of TTBCR.EAE has the value 0, selecting use of the Short-descriptor translation table format for PL1&0 stage 1 translations.
5) Software executes an ATS1C* operation to an address which, if accessed from Non-secure PL1, is associated with a domain configured as "Manager" or "No access".

or:

1) The processor is executing in Hyp mode.
2) HSCTLR.M has the value 1, enabling the PL2 stage 1 MMU.
3) The Non-secure copy of SCTLR.M has the value 0, disabling the PL1&0 stage 1 MMU.
4) The Non-secure copy of TTBCR.EAE has the value 0, selecting use of the Short-descriptor translation table format for PL1&0 stage 1 translations.
5) The Non-secure copy of DACR.D0 has the value 0b00 or 0b10, configuring domain 0 as "No access" or using the reserved encoding.
6) Software executes an ATS1C* operation.

If the first set of conditions is met then the value written to the PAR will be as if the associated domain for the address were configured as "Client".

If the second set of conditions is met then the value written to the PAR will erroneously indicate a domain fault.

Implications

Software running in Hyp mode with the PL2 stage 1 MMU disabled can receive incorrect results in the PAR for ATS1C* operations where the PL1&0 stage 1 MMU is enabled. ARM expects that most software executing in Hyp mode will operate with the PL2 stage 1 MMU enabled, and therefore will be unaffected by the first set of conditions described in this erratum.

Software running in Hyp mode with the PL2 stage 1 MMU enabled will get erroneous domain faults indicated in the PAR for ATS1C* operations where the PL1&0 stage 1 MMU is disabled.

Workaround

Software running in Hyp mode can check the values of HSCTLR.M and the Non-secure copy of SCTLR.M to determine whether they are different. If the values differ, it might be necessary for the Hyp mode software to emulate the ATS1C* operations by directly examining the stage 1 page tables. If the Non-secure copy of SCTLR.M has the

value 0 then there is a direct mapping between the virtual address and the intermediate physical address required as the input address for the stage 2 translation.

802022: A CPU can interfere with the duplicate tag RAM invalidation process for another CPU and cause deadlock

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The Cortex-A7 MPCore processor supports memory coherency between multiple CPUs within the processor, using a duplicate copy of the CPU L1 data tag RAMs in the SCU to filter coherency traffic. The CPU L1 data tag RAMs and duplicate tag RAMs are synchronized by holding **L1RSTDISABLE** LOW when each CPU is powered up for the first time after full processor power down. The synchronization cannot be performed using software. Because of this erratum, under certain power-up scenarios, the last set of the duplicate tag RAMs can remain inconsistent with the CPU L1 tag RAMs, leading to a deadlock.

Configurations Affected

This erratum does not affect the processor if one of the following is true:

- The variant and revision reported by the MIDR is r0p3 and REVIDR[0] is set to 1.
- The variant and revision reported by the MIDR is r0p4 and REVIDR[0] is set to 1.

To be affected by this erratum, the processor must be implemented with more than one CPU.

Conditions

1) The Cortex-A7 MPCore processor is powered down, so that the state of the L1 duplicate tag RAMs is corrupted.
2) The processor is powered up.
3) One of the CPUs within the processor, CPU A, is powered up (possibly at the same time as the processor). Another CPU, CPU B, remains powered down.
4) CPU A performs a cacheable data read, either explicitly due to a load or implicitly due to a prefetch or other cause, that misses in the L1 cache, causing an L1 linefill.
5) CPU B is powered up, causing an invalidation of the L1 duplicate tags for that CPU.
6) Data is returned for the L1 linefill for CPU A.
7) The L1 duplicate tags corresponding to the linefill for CPU A are updated in the same cycle that the tag invalidation sequence for CPU B reaches the last set of the L1 duplicate tag RAMs.

If these conditions are met, then the last set of the L1 duplicate tag RAMs for CPU B can be left in a corrupt state. This can lead to the lines in the last set being marked as valid. When CPU B then tries to allocate a line into the last set of the L1 cache, this corrupt data can be interpreted as a valid address. This address can be allocated into a buffer within the processor, but since this address was not valid in the L1 cache, the buffer is never deallocated. When CPU B next issues an operation which requires this buffer, it will wait indefinitely, causing a deadlock.

Implications

This erratum can only affect systems that use a power strategy where the entire Cortex-A7 MPCore processor is powered down, and then on power-up only one CPU is powered up, with other CPUs being powered up only after the first CPU has started executing. In affected systems, the processor can deadlock.

Workaround

The system software must ensure that active CPUs do not perform L1 data cache allocations during the tag invalidation process of other CPUs. This is only necessary the first time each CPU is brought out of reset after full processor power down. Three methods to ensure this are described below.

Method 1

This method coordinates the tag invalidation process for all CPUs. CPUs that are not required can then be powered down again using the standard power-down sequence. The CPUs can be powered up individually at a later time.

1) On processor power-up power is applied to all CPUs.
2) All CPUs are brought out of reset at the same time.

3) All CPUs complete the normal power-up sequence.

4) CPUs that are not required complete the normal power-down sequence.

Method 2

This method is applicable if power control software uses one CPU to initiate power up of other CPUs. In this method, CPU A powers up all other CPUs during CPU A's power-up sequence. CPUs that are not required can then be powered down again using the normal power-down sequence. The CPUs can be powered up individually at a later time.

CPU A

Initial state: powered down

1) Begin power-up sequence. Do not change SCTLR.C from its reset value of 0. This prevents allocations into the L1 data cache.

2) Clear the 'complete' flag for the power-up process, and clear the 'invalidation done' flags for other CPUs. Flags must be located in separate memory locations so they can be set and cleared individually using non-overlapping stores.

3) Execute a DSB instruction.

4) Initiate power-up of other CPUs.

5) Execute a WFE instruction.

6) On wake-up, check that 'invalidation done' flags have been set by all other CPUs. If a flag is not set, return to WFE state and repeat this step.

7) Set the 'complete' flag.

8) Execute a DSB instruction.

9) Continue the normal power-up sequence. This includes setting SCTLR.C to 1 to enable allocations into the L1 data cache.

Other CPUs

Initial state: powered down

1) Begin normal power-up sequence.

2) Perform a TLBIMVAIS operation to any address.

3) Execute a DSB instruction. The previous TLBIMVAIS operation requires the DSB to send a DVM Synchronization request. The request will stall until the L1 duplicate tag invalidation process is complete.

4) Set the 'invalidation done' flag.

5) Execute a DSB instruction.

6) Execute a SEV instruction.

7) Execute a WFE instruction.

8) On wake-up, check that the 'complete' flag has been set by CPU A. If the flag is not set, return to WFE state and repeat this step.

9) Continue the normal power-up sequence.

10) Perform the normal power-down sequence if the CPU is not required.

Method 3

This method is applicable if power control software uses one CPU to initiate power up of other CPUs, and can send interrupts to all active CPUs. In this method, a CPU is only powered up when it is required. Active CPUs are put into WFE during the power-up sequence of the new CPU.

CPU A

Initial state: powered up

1) Clear the 'complete' flag for the new CPU power-up process, clear the 'ready' flags for each CPU that is already powered-up, and clear the 'invalidation done' flag for the CPU to be powered-up. Flags must be located in separate memory locations so they can be set and cleared individually using non-overlapping stores.

2) Execute a DSB instruction.

3) Send interrupt to active CPUs.

4) Clear SCTLR.C to 0. This waits for outstanding L1 data cache allocations to complete and then prevents further allocations. The cache contents are preserved and cacheable loads and stores continue to lookup in the cache.

5) Execute an ISB instruction.
6) Execute a DSB instruction.
7) Execute a WFE instruction.
8) On wake-up, check that 'ready' flags have been set by all active CPUs. If a flag is not set, return to WFE state and repeat this step.
9) Initiate power-up of the new CPU.
10) Execute a WFE instruction.
11) On wake-up, check that the 'invalidation done' flag has been set by the new CPU. If the flag has not been set, return to WFE state and repeat this step.
12) Set the 'complete' flag.
13) Execute a DSB instruction.
14) Execute a SEV instruction.
15) Set SCTLR.C to 1.
16) Execute an ISB instruction.
17) Execute a DSB instruction.
18) Continue normal execution.

Other active CPUs

Initial state: powered up

1) Receive interrupt from CPU A.
2) Clear SCTLR.C to 0. This waits for outstanding L1 data cache allocations to complete and then prevents further allocations. The cache contents are preserved and cacheable loads and stores continue to lookup in the cache.
3) Execute an ISB instruction.
4) Execute a DSB instruction.
5) Set the 'ready' flag.
6) Execute a DSB instruction.
7) Execute a SEV instruction.
8) Execute a WFE instruction.
9) On wake-up, check that the 'complete' flag has been set by CPU A. If the flag is not set, return to WFE state and repeat this step.
10) Set SCTLR.C to 1.
11) Execute an ISB instruction.
12) Execute a DSB instruction.
13) Continue normal execution.

New CPU

Initial state: powered down

1) Perform the normal power-up sequence.
2) Perform a TLBIMVAIS operation to any address.
3) Execute a DSB instruction. The previous TLBIMVAIS operation requires the DSB to send a DVM Synchronization request. The request will stall until the L1 duplicate tag invalidation process is complete.
4) Set the 'invalidation done' flag.
5) Execute a DSB instruction.
6) Execute a SEV instruction.
7) Continue normal execution.

814220: Cache maintenance by set/way operations can execute out of order

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

The v7 ARM ARM states that all cache and branch predictor maintenance operations that do not specify an address execute, relative to each other, in program order. However, because of this erratum, an L2 set/way cache maintenance operation can overtake an L1 set/way cache maintenance operation.

Configurations Affected

To be affected by this erratum, the processor must be implemented with an L2 cache.

Conditions

For this erratum to have an observable effect, the following conditions must be met.

1) A CPU performs an L1 DCCSW or DCCISW operation.
2) The targeted L1 set/way contains dirty data.
3) Before the next DSB, the same CPU executes an L2 DCCSW or DCCISW operation while the L1 set/way operation is in progress.
4) The targeted L2 set/way is within the group of L2 set/ways that the dirty data from L1 can be allocated to.

If the above conditions are met then the L2 set/way operation can take effect before the dirty data from L1 has been written to L2.

Note: Conditions (3) and (4) are not likely to be met concurrently when performing set/way operations on the entire L1 and L2 caches. This is because cache maintenance code is likely to iterate through sets and ways in a consistent ascending or consistent descending manner across cache levels, and to perform all operations on one cache level before moving on to the next cache level. This means that, for example, cache maintenance operations on L1 set 0 and L2 set 0 will be separated by cache maintenance operations for all other sets in the L1 cache. This creates a large window for the cache maintenance operations on L1 set 0 to complete.

Implications

Code that intends to clean dirty data from L1 to L2 and then from L2 to L3 using set/way operations might not behave as expected. The L2 to L3 operation might happen first and result in dirty data remaining in L2 after the L1 to L2 operation has completed.

If dirty data remains in L2 then an external agent, such as a DMA agent, might observe stale data.

If the processor is reset or powered-down while dirty data remains in L2 then the dirty data will be lost.

Workaround

Correct ordering between set/way cache maintenance operations can be forced by executing a DSB before changing cache levels.

844169: Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

The ARMv7 architecture requires that when all associated stages of translation are disabled for the current privilege level, memory locations are only accessed due to instruction fetches within the same or next 4KB region as an instruction which has been or will be fetched due to sequential execution. In the conditions detailed below, the Cortex-A7 MPCore processor might access other locations speculatively due to instruction fetches.

Configurations affected

All configurations of Cortex-A7 are affected.

Conditions

1) The processor must be executing at PL2 or Secure PL1.
2) Address translation is disabled for the current exception level (by clearing the appropriate SCTLR.M or HSCTLR.M bit).
3) The HCR.VM bit is set.

Implications

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

Workaround

Because the HCR.VM bit is reset low, this situation is most likely to arise in powerdown code, if PL2 or Secure PL1 software disables address translation before the core is powered down. To work around this erratum, software should ensure that HCR.VM is cleared before disabling address translation at PL2 or Secure PL1.

856125: Stage 2 XN attribute is suppressed when stage 1 MMU is disabled

Category B

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

The Cortex-A7 MPCore processor implements the ARM Virtualization Extensions. The Virtualization Extensions provide independent translation regimes for memory accesses from different modes. In the Non-secure PL1&0 translation regime, address translation occurs in two stages. Stage 1 maps the Virtual Address (VA) to an Intermediate Physical Address (IPA). Stage 2 maps the IPA to the Physical Address (PA).

The ARMv7-A architecture states that if the stage 2 Execute-never (XN) attribute is set to 1, execution from the region is not permitted, regardless of the value of the XN attribute in the stage 1 translation. If a Permission fault is generated because the stage 2 XN bit is set to 1, this is reported as a stage 2 MMU fault.

Because of this erratum, the stage 2 XN attribute is suppressed when the stage 1 MMU is disabled.

Configurations Affected

All configurations.

Conditions

1) The PL1&0 stage 1 MMU is disabled (SCTLR.M is set to 0).
2) The PL1&0 stage 2 MMU is enabled (HCR.VM is set to 1).
3) The CPU is operating in Non-secure state at the PL0 or PL1 privilege level.

Implications

There are two main implications of this erratum:

1) For code running at Non-secure PL1 in a CPU with the stage 1 MMU disabled, as might occur during the PL1 initialization of a CPU under virtualization, the stage 2 XN attribute does not give any protection from speculative instruction fetches from read-sensitive locations. These locations might rarely be corrupted because of this erratum.
2) For code running at PL0 with HCR.TGE == 1, which permits running an application directly on a hypervisor, the stage 2 XN permission cannot be used to protect areas of memory from being executed. This means that enforcing a security policy in which areas that are writeable are not executable is not possible at EL0 with HCR.TGE == 1.

Workaround

The first implication can be worked around by ensuring that no read-sensitive locations are mapped into the stage 2 page tables or have read access in the stage 2 page tables when running at Non-secure EL1 with the MMU disabled.

The second implication cannot be worked around while HCR.TGE == 1, but it does not lead to functional misbehavior of correctly constructed code.

2.6. Category B (Rare)

There are no errata in this category

2.7. Category C

777869: DBGDSCRext[15:14] is masked when DBGEN is LOW and OS Lock is SET

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2

Description

The Cortex-A7 MPCore processor includes support for external debug over powerdown as required by the v7.1 Debug Architecture. As part of the OS Save and Restore mechanism, software should be able to read the programmed value of **DBGDSCRext[15:14]** when the OS Lock is set. However, due to this erratum, **DBGDSCRext[15:14]** is RAZ if invasive debug is disabled when the OS Lock is set.

Configurations Affected

All configurations in which invasive debug is not permanently disabled.

Conditions

1) Invasive debug is disabled and the OS Lock is set.
2) **DBGDSCRext** is read using any of the debug register interfaces.

Implications

If invasive debug is disabled, when software saves the debug registers prior to a powerdown, it reads and saves zero instead of the programmed value of **DBGDSCRext[15:14]**. Consequently, when software restores the debug registers after powerdown, it leaves both Monitor and Halting debug-modes disabled rather than in the same state as prior to powerdown.

Workaround

Ensure invasive debug is enabled when saving the value of the **DBGDSCR** during OS Save and Restore.

781169: DBGPRSR[1:0] are incorrectly RAZ during debug over powerdown

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2

Description

The Cortex-A7 MPCore processor includes support for external debug over powerdown as required by the v7.1 Debug Architecture. The architecture states that the value 0b00 for **DBGPRSR[1:0],** indicating certain of the debug registers cannot be accessed but have not lost their value, is not permitted. However, due to this erratum, a read of these bits during debug over powerdown will return the value 0b00.

Configurations Affected

To be affected by this erratum, all of the following must apply:

- The processor is built with the integration layer.
- The system makes use of debug over powerdown functionality.

Conditions

1) An external debugger reads **DBGPRSR** when the processor is powered down.

Implications

When the processor is powered down, the value returned by a read of **DBGPRSR[1]** incorrectly indicates the processor has not powered down since the last time this register was read.

Workaround

When reading **DBGPRSR[1:0]**, an external debugger must interpret a value of 0b00 as 0b10. The value 0b10 indicates that the processor is powered down and some debug registers cannot be accessed.

784472: CTI Authentication Status register is incorrect

Category C
Products Affected: Cortex-A7 MPCore.
Present in: r0p2

Description

The AUTHSTATUS register is a read-only register in the CTI that reports the required security state for debug operations, and the current values of the debug enable signals.

The CoreSight Architecture Specification specifies bits [3:0] in the AUTHSTATUS register as follows:

- [3:2] NSNID, Non-Secure Non-Invasive Debug
- [1:0] NSID, Non-Secure Invasive Debug

For each of these fields, the permitted values of the field are:

0b10 Functionality disabled

0b11 Functionality enabled

In the CTI, the AUTHSTATUS{NSNID, NSID} fields currently read:

- When functionality is disabled - 0b01

As indicated earlier, this should read 0b10. That is, the two bits are reversed.

Configurations Affected

This erratum affects implementations of the processor that include the integration layer.

Condition 1

The following conditions must apply when reading AUTHSTATUS[1:0] - Non-secure Invasive Debug

- The **DBGEN** input to the CTI is LOW
- The AUTHSTATUS register is read

Condition 2

The following conditions must apply when reading AUTHSTATUS[3:2] - Non-secure non-Invasive Debug

- The **NIDEN** input to the CTI is LOW
- The **DBGEN** input to the CTI is LOW
- The AUTHSTATUS register is read

Implications

The status of the CTI debug authentication signals returned by the AUTHSTATUS register read is incorrect. The masking of trigger inputs and outputs by **DBGEN** and **NIDEN** is not affected by this erratum. However, the return of an incorrect value might lead to incorrect operation of debug tools.

Workaround

This is a workaround for users and tools vendors.

When reading the AUTHSTATUS register, swap the bits in the affected fields and interpret the returned data accordingly.

784519: DBGPRCR.CORENPDRQ incorrectly changed on warm reset

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2

Description

The Cortex-A7 MPCore processor implements the v7.1 Debug Architecture. This specifies that the DBGPRCR.CORENPDRQ bit should be reset by a core powerup reset only, but because of this erratum it is also reset by a warm reset.

Configurations Affected

To be affected by this erratum, the processor must be implemented in a system that supports emulation of powerdown.

Conditions

1) The values of the DBGPRCR.{COREPURQ, CORENPDRQ} bits differ.
2) A warm reset occurs.

Implications

In a system which is affected by this erratum, the Cortex-A7 MPCore processor might report its powerdown emulation request status incorrectly to the power controller after a warm reset.

Workaround

If a workaround is required, the nature of the workaround depends on which debug solution is used. If an external debugger is used, it should keep the DBGPRCR.COREPURQ bit asserted for as long as emulation of powerdown is required. If self-hosting software is used, its warm reset handler should set the DBGPRCR.CORENPDRQ bit whenever emulation of powerdown is required.

790569: Accesses to debug data transfer registers when the OS Lock is set behave incorrectly

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

The Cortex-A7 MPCore processor includes support for external debug over powerdown as required by the v7.1 Debug Architecture. As part of the OS Save and Restore mechanism, the debug data transfer registers should have certain behaviors when the OS Lock is set. However, because of this erratum, the debug data transfer registers do not have the required behaviors when the OS Lock is set.

Configurations Affected

All configurations.

Conditions

1) OS Lock is set.
2) DBGDTRRXext or DBGDTRTXext is read or written.

Implications

Any state relating to the debug data transfer registers cannot be preserved over powerdown by the OS Save and Restore mechanism. Note the erratum fix only fully corrects the behavior of the debug data transfer registers for the system instructions used by the OS Save and Restore mechanism. The erratum fix incorrectly allows memory-mapped interface accesses to debug data transfer registers to have side effects when the OS Lock is set. However, the expected use model is to use system instructions for OS Save and Restore, and therefore there is no expected problem due to the incompleteness of the erratum fix.

Workaround

There is no workaround for this erratum.

790570: Accesses to debug registers when the OS Double Lock is set may result in incorrect behavior

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

The Cortex-A7 MPCore processor includes support for external debug over powerdown as required by the v7.1 Debug Architecture. As part of the OS Save and Restore mechanism, accesses to most external debug registers are restricted when the OS Double Lock is set. However, because of this erratum, an external access might not result in an error or a write might still occur when an error is returned.

Configurations Affected

All configurations.

Conditions

1) OS Double Lock is set.

2) An external debug access occurs.

Implications

External writes to debug registers when the OS Double Lock is set might still cause side effects. For example, an attempt to write 1 to DBGPRSR.CWRR when the OS Double Lock is set might cause a core warm reset.

Workaround

If the OS Double Lock is set on a core in a processor built with the integration layer, the DBGPWRDUP pin for that core should be deasserted LOW. This ensures debug accesses are handled by the debug over powerdown logic in the integration layer rather than by the debug logic in the core.

791620: A non-cacheable store in a tight loop might not become observable until the loop completes

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

The ARM architecture states that all writes complete in a finite period of time in implementations that include the Multiprocessing Extensions. This applies for all writes, including repeated writes to the same location. Because of this erratum, an uninterrupted pattern of stores to the same non-cacheable 64-byte aligned memory region might not become observable to another CPU or master in the system.

Configurations Affected

This erratum affects all configurations of the Cortex-A7 MPCore processor.

Conditions

1) CPU A executes a loop containing one or more stores to the same Normal Non-Cacheable 64-byte aligned memory region.
2) The loop contains fewer than two consecutive instructions that are not stores to the same 64-byte aligned memory region.
3) The loop does not contain any of the following:
   A. A load to the same 64-byte aligned memory region
   B. A store to a different 64-byte aligned memory region in Normal Non-Cacheable, Device, or Strongly Ordered memory
   C. A DMB or DSB instruction
   D. A Load-Exclusive, Store-Exclusive, or CLREX instruction
   E. A WFE or WFI instruction
   F. A CP15 cache or TLB maintenance operation

If the above conditions are met, then the repeated stores to the same location might continuously merge inside the CPU until the loop completes on CPU A.

Implications

Another CPU in the processor or another master in the system that reads from the 64-byte aligned memory region might not receive the newest data until the loop completes on CPU A.

The erratum is not expected to be observed in real code for the following reasons:

- System timers and interrupts will normally change the program flow on CPU A long enough for the stores to become observable.
- The last store will become observable to other CPUs and masters when the loop completes on CPU A.
- Polled variables that are being updated in a loop are likely to contain a barrier or a power-saving measure such as WFE or WFI.
- Loops will normally contain sufficient instructions between stores to the same 64-byte aligned memory region to avoid this issue.

Workaround

A workaround is not expected to be necessary in real code. However, if a workaround is required then the CPU executing the loop can insert any of the instructions mentioned in condition (3) to avoid the erratum.

If the software on CPU A cannot be modified then the recommended workaround is to force CPU A to regularly take an interrupt which would act as a watchdog. Several options are possible to generate this regular interrupt, which might be specific to each system. Interrupts generated by the local timer, global timer, or PMU cycle counter overflow are possible candidates.

792169: A read of DBGOSLSR during powerdown returns an incorrect value

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

The Cortex-A7 MPCore processor includes support for external debug over powerdown as required by the v7.1 Debug Architecture. Because of this erratum, the value returned by a read of the OS Lock Status Register, DBGOSLSR[3:0], is incorrect during powerdown.

Configurations Affected

To be affected by this erratum, all of the following must apply:

- The processor is built with the integration layer. The system makes use of debug over powerdown functionality.

Conditions

1) An external debugger reads DBGOSLSR[3:0] when the processor is powered down.

Implications

When the processor is powered down, a read of DBGOSLSR[3:0] returns incorrect information.

The OS Lock Model implementation field, DBGOSLSR.OSLM, reads as 0b11. This is a reserved value with no meaning. It should read as 0b10 to indicate that OS Lock is implemented and DBGOSSRR is not implemented.

The not 32-bit access field, DBGOSLSR.nTT, reads as 0b1. It should read as 0b0 to indicate that a 32-bit access is needed to write the key to the OS Lock Access Register.

Workaround

When reading DBGOSLSR[3:0], an external debugger must interpret a value of 0b1111 as 0b10x0, where DBGOSLSR.OSLK is UNKNOWN.

793369: Loads and stores with mismatched attributes might cause deadlock

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

If the translation tables are constructed such that two or more different virtual addresses map to the same physical address, but the virtual addresses have different memory types, then under certain conditions the processor might deadlock.

Configurations Affected

All configurations.

Conditions

The following conditions must be met, in the order shown, by a single CPU in the processor.

1) The CPU executes a store instruction to Normal memory. The Shareability attribute of the virtual address must be shareable, and the Cacheability attributes must include one of the following:
   A. Inner Write-Through
   B. Inner Non-Cacheable and Outer Write-Back or Write-Through
2) The CPU executes a Load Multiple that includes loads to the same cacheline as the store. The Inner Cacheability attribute of the virtual address for this Load Multiple must include Write-Back cacheable.
3) A load in the Load Multiple misses in the L1 cache, causing the cacheline to be allocated before the store from condition 1 completes.

Implications

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

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

Workaround

If multiple aliases to the same physical page in memory are required, place a memory barrier instruction between load or store instructions to the aliased locations.

794322: An instruction fetch can be allocated into the L2 cache after the cache is disabled

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3

Description

The SCTLR.C bit controls when data can be allocated to data and unified caches. On a Cortex-A7 processor, instruction fetches can cause lines to be allocated to the L2 cache when the C bit is set to 1. Because of this erratum, an instruction fetch might cause an allocation to the L2 cache when the C bit is set to 0.

Configurations Affected

This erratum affects configurations of the Cortex-A7 MPCore processor with an L2 cache.

Conditions

1) The CPU executes some code while the SCTLR.I and SCTLR.C bits are both set to 1.
2) An instruction cache linefill starts to a 32-byte aligned memory region because of the code being executed. This prompts an L2 linefill to the 64-byte aligned memory region that contains the data required by the instruction cache linefill.
3) The SCTLR.C bit is cleared to 0 to disable the L2 cache.
4) A sequence of DCCSW, DCISW, or DCCISW operations is used to clean or invalidate the entire L1 cache.
5) A sequence of DCCSW, DCISW, or DCCISW operations is used to clean or invalidate the entire L2 cache.
6) The CPU does not attempt to execute code at any address in the last two beats of L2 linefill data before the matching index in the L2 cache is cleaned or invalidated by the sequence of cache maintenance operations.
7) The index in the L2 cache corresponding to the L2 linefill address is cleaned or invalidated by the sequence of cache maintenance operations before the last beat of L2 linefill data is returned from the interconnect.
8) Between disabling the cache, and cleaning or invalidating the cache, the software does not perform a TLB maintenance operation followed by a DSB instruction.

If the above conditions are met, and DCISW or DCCISW instructions are used, then after the sequence the L2 cache might incorrectly contain one valid cache line.

If the above conditions are met, and DCCSW or DCCISW instructions are used, and the instruction linefill accessed shareable memory, and the interconnect returned dirty data for the linefill, then after the sequence the L2 cache might incorrectly contain one valid cache line with dirty data.

Implications

A clean or invalidate of the entire L2 cache is typically used as part of a powerdown sequence. If the L2 cache contains a dirty line when the processor is powered down, then the data in that line is lost. If the L2 cache contains a clean line when the processor is powered down then no data is lost.

Cleaning or invalidating an entire cache is an operation that takes a long time. Depending on the cache size, this can be many thousands of cycles. This erratum requires the response from the interconnect for the instruction linefill to take longer than the L1 cache clean or invalidate, which is unlikely to happen in most systems.

Workaround

This erratum can be avoided by inserting both of the following after the SCTLR.C bit is cleared to 0, and before the caches are cleaned or invalidated:

1) A TLBIMVA operation to any address.
2) A DSB instruction.

799972: Non-cacheable request event incorrectly counts cacheable instruction accesses

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The Cortex-A7 MPCore processor implements version 2 of the Performance Monitor Unit architecture (PMUv2). The PMU provides the ability to gather various statistics on the operation of the processor and memory system during runtime. This event information can be used when debugging or profiling code. Because of this erratum, the Non-cacheable external memory request event provides inaccurate information.

Configurations Affected

All configurations.

Conditions

1) A PMU counter is enabled and programmed to count Non-cacheable external memory requests (event 0xC1).

2) Cacheable instruction fetches miss in the L1 instruction cache, and start L1 linefills.

When the above conditions are met, the event counter will increment for each instruction cache linefill, in addition to Non-cacheable instruction and data fetches.

Implications

A PMU counter that is programmed to count Non-cacheable external memory requests might not provide accurate information. The information returned can be misleading when debugging or profiling code executed on the processor.

Workaround

Enable a second PMU counter and program it to count instruction fetches that cause linefills (event 0x01). Subtract the value returned by this counter from the value returned by the Non-cacheable external memory request counter (event 0xC1). The result of the subtraction is the number of Non-cacheable external memory requests.

800721: Reads of the DBGPCSR can cause incorrect values to be latched into DBGCIDSR and DBGVIDSR

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The Cortex-A7 MPCore processor supports sample-based profiling of software. Reads of the DBGPCSR register return the address of a recently-executed instruction, and cause information about the context of the instruction to be latched into DBGCIDSR and DBGVIDSR. Because of this erratum, in some cases the information captured in the DBGCIDSR and DBGVIDSR can be incorrect.

Configurations Affected

All configurations.

Conditions

1) The processor is executing in a state where non-invasive debug is permitted.
2) The processor transitions between Secure and Non-Secure state, or into or out of Hyp mode.
3) Before the processor executes the first instruction after this transition, the DBGPCSR is read.

If these conditions are met, then the values sampled into the DBGCIDSR and DBGVIDSR will reflect the new context of the processor, not the context in which the sampled instruction was executed.

If the processor transitioned from Non-Secure to Secure state and non-invasive debug is not permitted in Secure state, and the DBGPCSR read is one cycle after the transition, then the DBGCIDSR will be sampled with the value of the secure CONTEXTIDR.

Implications

In cases where sample-based profiling is in use, this erratum can reduce the accuracy of the data.

If Secure non-invasive debug is not permitted, then there is a one-cycle window where a Non-secure read could observe the value of the secure CONTEXTIDR. It is not possible for reads issued by the same processor to observe this value, only another processor or agent in the system can perform a read with the required timing.

Workaround

There is no workaround for this erratum.

802119: Debug registers DBGLSR, DBGDEVID, and DBGDEVID1 are only accessible during powerdown if DBGSWENABLE[CPU] is asserted

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The Cortex-A7 MPCore processor includes support for external debug over powerdown as required by the v7.1 Debug Architecture. The v7.1 Debug Architecture states that access to debug registers through the external debug interface is unaffected by the Debug Software Enable function. However, because of this erratum, DBGLSR, DBGDEVID, and DBGDEVID1 are only accessible during powerdown if **DBGSWENABLE[CPU]** is asserted.

Configurations Affected

To be affected by this erratum, all of the following must apply:

- The processor is built with the integration layer.
- The system makes use of debug over powerdown functionality.

Conditions

1) **DBGSWENABLE[CPU]** is not asserted.
2) The processor is powered down.
3) An external debugger reads DBGLSR, DBGDEVID, or DBGDEVID1.

Implications

When the conditions are met, the value returned by the read of DBGDEVID, DBGDEVID1, or DBGLSR is zero.

For DBGDEVID, this means the read value incorrectly indicates:

- DBGEACR, DBGOSDLR, DBGCIDSR, and DBGVIDSR are not present.
- DBGPCSR is not implemented as register 40.
- The processor does not implement the Virtualization Extensions.
- Breakpoint address masking might be implemented.
- Watchpoint address masking might be implemented.

Architecturally, the correct read value would be 0x01110F13, which indicates:

- DBGEACR and DBGOSDLR are present.
- DBGPCSR is implemented as register 40.
- DBGCIDSR is implemented as register 41.
- DBGVIDSR is implemented as register 42.
- The processor implements the Virtualization Extensions.
- Breakpoint address masking is not implemented.
- Watchpoint address masking is implemented.

For DBGDEVID1, the read value incorrectly indicates that DBGPCSR samples are offset by a value that depends on the instruction state. Architecturally, the correct read value would be 0x00000001, which indicates that no offset is applied to the DBGPCSR samples.

For DBGLSR, zero is the correct read value for an external debug access. Therefore, the effect of DBGSWENABLE[CPU] is only observable if the external debug access is not identified as an external debug access. That is, if PADDRDBG[31] is set to 0 (system) instead of 1 (external debugger). In this scenario, the DBGLSR read value might incorrectly indicate that the Software Lock is clear when it is set. In addition, DBGLSR.SLI, which should be RAO to indicate that Software Lock is implemented, will instead be RAZ.

Workaround

The external debugger should read the Main ID Register, MIDR, to determine which processor it is accessing. It should use this information to determine the correct read value for DBGDEVID and DBGDEVID1.

Software that makes use of the Software Lock must keep track of whether it has set or cleared the lock. It must not rely on the value returned when DBGLSR is read during powerdown.

803219: TBB/TBH incorrect branch address is not mis-predicted

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The TBB and TBH instructions cause a PC-relative forward branch using a table of single byte or halfword offsets. A base register provides a pointer to the table, and a second register supplies an index into the table. Because of this erratum, the processor might incorrectly calculate the branch offset used to determine whether a predicted branch is correct. In this event, a branch with an address mis-prediction might be treated as a correct branch and cause the processor to execute code from the wrong address.

Configurations Affected

All configurations.

Conditions

1) A load or store instruction with pre-indexed base-register writeback is issued.
2) A TBB or TBH instruction is issued in the cycle after the load or store instruction with the same base-register as the preceding load or store instruction.
3) The TBB or TBH instruction is predicted taken.
4) The load or store instruction and the TBB or TBH instruction pass their condition code checks.
5) The predicted target address is equal to the base-register plus the offset loaded from memory.

If the above conditions are met then the processor branches to the base-register plus offset rather than the PC plus offset.

Example code sequence:

```
LDR r8, [r0, #0x4]!
<independent instruction>
TBB [r0, r5]
```

Where <independent instruction> can be one of:

- No instruction
- NOP
- IT
- B, BL, BLX (imm)
- MOV (reg)
- MOV (imm), ADD (imm), SUB (imm), CMP (imm), CMN (imm), AND (imm), EOR (imm), TST (imm), TEQ (imm), BIC (imm), ORR (imm), MVN (imm), ORN (imm), UXTB, UXTB16, UXTH, SXTB, SXTB16, SXTH

Implications

The processor will execute code from the wrong address if permissions are correct.

The erratum will not cause the processor to change privilege level or security state.

Note: These implications can be safely ignored for compiled C/C++ code because the branch table is opaque to the programmer and C/C++ compilers will not generate the code sequence described in conditions (1) to (4). Similarly, the code sequence is not expected to be generated by other compilers or interpreters. In addition, the 32-bit result of the calculation in condition (5) is unlikely to match the branch target address predicted by the BTAC. This erratum has not been observed in real code.

Workaround

There is no workaround for this erratum.

803269: System instruction accessing CNTPCT or CPACR might be ignored in debug state

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4

Description

The Cortex-A7 MPCore processor includes the Virtualization Extensions, which support trapping system instructions to the hypervisor under certain conditions. Hypervisor traps should be ignored in debug state, but due to this erratum system instructions that access the CNTPCT or CPACR are ignored in debug state if the conditions for generating a hypervisor trap would be met if the processor was not in debug state.

Configurations Affected

All configurations.

Conditions

For CNTPCT:

1) The Cortex-A7 MPCore processor is in debug state.
2) The Cortex-A7 MPCore processor is in a Non-secure mode other than Hyp mode.
3) CNTHCTL.PL1PCTEN is set to 1.
4) A system instruction that accesses CNTPCT is executed.

For CPACR:

1) The Cortex-A7 MPCore processor is in debug state.
2) The Cortex-A7 MPCore processor is in a non-secure mode other than Hyp mode.
3) HCPTR.TCPAC is set to 1.
4) A system instruction that accesses CPACR is executed.

Implications

A system instruction that accesses the CNTPCT or the CPACR in non-secure state in debug state might have no effect.

Workaround

When in debug state, the relevant trap bit (CNTHCL.PL1PCTEN for the CNTPCT and HCPTR.TCPAC for the CPACR) must be cleared to zero before trying to access the CNTPCT or the CPACR.

804069: Exception mask bits are cleared when an exception is taken in Hyp mode

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

The Cortex-A7 MPCore processor implements the ARM Virtualization Extensions and the ARM Security Extensions. Exceptions can be routed to Monitor mode by setting SCR.{EA, FIQ, IRQ} to 1. Exceptions can be masked by setting corresponding bit CPSR.{A, I, F} to 1.

The ARMv7-A architecture states that an exception taken in Hyp mode does not change the value of the mask bits for exceptions routed to Monitor mode. However, because of this erratum, the corresponding mask bits will be cleared to 0.

Configurations Affected

All configurations.

Conditions

1) One or more exception types are routed to Monitor mode by setting one or more of SCR.{EA, FIQ, IRQ} to 1.
2) The corresponding exception types are masked by setting the corresponding CPSR.{A, F, I} bits to 1.
3) Any exception is taken in Hyp mode.

If the above conditions are met then the exception mask bit CPSR.{A, F, I} is cleared to 0 for each exception type that meets conditions (1) and (2). The affected mask bits are cleared together regardless of the exception type in condition (3).

Implications

If SCR.{AW, FW} is set to 0 then the clearing of corresponding bit CPSR.{A, F} to 0 has no effect. The value of CPSR.{A, F} is ignored.

Otherwise, when CPSR.{A, F, I} is set to 1, Secure code cannot rely on CPSR.{A, F, I} remaining set to 1. An exception that should be masked might be routed to Monitor mode.

This is Category C as it is expected that users will:

1) set SCR.{AW, FW} to 0 when SCR.{EA, FIQ} is set to 1.
2) set SCR.IRQ to 0.

Workaround

There is no workaround for this erratum.

805420: PMU event counter 0x14 does not increment correctly

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

The Cortex-A7 MPCore processor implements version 2 of the Performance Monitor Unit architecture (PMUv2). The PMU can gather statistics on the operation of the processor and memory system during runtime. This event information can be used when debugging or profiling code. When a PMU counter is programmed to count L1 instruction cache accesses (event 0x14), the counter should increment on all L1 instruction cache accesses. Because of this erratum, the counter increments on cache hits but not on cache misses.

Configurations Affected

All configurations.

Conditions

1) A PMU counter is enabled and programmed to count L1 instruction cache accesses (event 0x14).

2) Cacheable instruction fetches miss in the L1 instruction cache.

When the above conditions are met, the event counter will not increment.

Implications

A PMU counter that is programmed to count L1 instruction cache accesses will count instruction cache hits but not instruction cache misses.

The information returned can be misleading when debugging or profiling code executed on the processor.

Cache-bound code execution is not affected by this erratum because of the absence of cache misses.

Workaround

To obtain a better approximation for the number of L1 instruction cache accesses, enable a second PMU counter and program it to count instruction fetches that cause linefills (event 0x01). Add the value returned by this counter to the value returned by the L1 instruction access counter (event 0x14). The result of the addition is a better indication of the number of L1 instruction cache accesses.

809719: PMU events 0x07, 0x0C, and 0x0E do not increment correctly

Category C

Products Affected: Cortex-A7 MPCore.

Present in: r0p2, r0p3, r0p4, r0p5

Description

The Cortex-A7 MPCore processor implements version 2 of the Performance Monitor Unit architecture (PMUv2). The PMU can gather statistics on the operation of the processor and memory system during runtime. This event information can be used when debugging or profiling code.

The PMU can be programmed to count architecturally executed stores (event 0x07), software changes of the PC (event 0x0C), and procedure returns (event 0x0E). However, because of this erratum, these events do not fully adhere to the descriptions in the PMUv2 architecture.

Configurations Affected

All configurations.

Conditions

Either:

1) A PMU counter is enabled and programmed to count event 0x07. That is: instruction architecturally executed, condition code check pass, store.
2) A PLDW instruction is executed.

If the above conditions are met, the PMUv2 architecture specifies that the counter for event 0x07 does not increment. However, the counter does increment.

Or:

1) A PMU counter is enabled and programmed to count event 0x0C. That is: instruction architecturally executed, condition code check pass, software change of the PC.
2) An SVC, HVC, or SMC instruction is executed.

If the above conditions are met, the PMUv2 architecture specifies that the counter for event 0x0C increments. However, the counter does not increment.

Or:

1) A PMU counter is enabled and programmed to count event 0x0E. That is: instruction architecturally executed, condition code check pass, procedure return.
2) One of the following instructions is executed:
    1. MOV PC, LR
    2. ThumbEE LDMIA R9!, {?, PC}
    3. ThumbEE LDR PC, [R9], #offset
    4. BX Rm, where Rm != R14
    5. LDM SP, {?, PC}

If the above conditions are met, the PMUv2 architecture specifies that the counter for event 0x0E increments for (a), (b), (c) and does not increment for (d) and (e). However, the counter does not increment for (a), (b), (c) and increments for (d) and (e).

Implications

The information returned by PMU counters that are programmed to count events 0x07, 0x0C, or 0x0E might be misleading when debugging or profiling code executed on the processor.

Workaround

There is no workaround for this erratum.