ARM Cortex^{TM}-A15 MPCore - NEON

Product Revision r3

Software Developers Errata Notice

Non-Confidential - Released

This document contains all software developer errata known at the date of issue in releases r3p0 up to and including revision r3p3

Copyright © 2015 ARM. All rights reserved.

Non Confidential

Software Developers Errata Notice

Copyright © 2015 ARM. All rights reserved.

Non-Confidential Proprietary Notice

This document is protected by copyright and the practice or implementation of the information herein may be protected by one or more patents or pending applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document.

This document is Non-Confidential but any disclosure by you is subject to you providing the recipient the conditions set out in this notice and procuring the acceptance by the recipient of the conditions set out in this notice.

Your access to the information in this document is conditional upon your acceptance that you will not use, permit or procure others to use the information for the purposes of determining whether implementations infringe your rights or the rights of any third parties.

Unless otherwise stated in the terms of the Agreement, this document is provided “as is”. ARM makes no representations or warranties, either express or implied, included but not limited to, warranties of merchantability, fitness for a particular purpose, or non-infringement, that the content of this document is suitable for any particular purpose or that any practice or implementation of the contents of the document will not infringe any third party patents, copyrights, trade secrets, or other rights. Further, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of such third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT LOSS, LOST REVENUE, LOST PROFITS OR DATA, SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Words and logos marked with ® or TM are registered trademarks or trademarks, respectively, of ARM Limited. Other brands and names mentioned herein may be the trademarks of their respective owners. Unless otherwise stated in the terms of the Agreement, you will not use or permit others to use any trademark of ARM Limited.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

Copyright © 2015 ARM Limited 110 Fulbourn Road, Cambridge, England CB1 9NJ. All rights reserved.

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-028093
- 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.

09 Oct 2015: Changes in Document v20

Page	Status	ID	Cat	Rare	Summary of Erratum
55	Updated	851024	Cat**C**		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores

21 Sep 2015: Changes in Document v19

Page	Status	ID	Cat	Rare	Summary of Erratum
55	New	851024	Cat**C**		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores

20 Feb 2015: Changes in Document v18

Page	Status	ID	Cat	Rare	Summary of Erratum
54	New	842119	Cat**C**		Instruction issued through ITR by debugger executes incorrectly for PCLKDBG frequencies much slower than the processor

04 Oct 2014: Changes in Document v17

Page	Status	ID	Cat	Rare	Summary of Erratum
32	New	836969	Cat**B**		Code sequence continuously hitting the L1 cache can block snoop

09 Sept 2014: Changes in Document v16

Page	Status	ID	Cat	Rare	Summary of Erratum
53	New	834569	CatC		PMU event BUS_CYCLES might be incorrect in some cases

08 Aug 2014: Changes in Document v15

Page	Status	ID	Cat	Rare	Summary of Erratum
31	New	830321	CatB		Cortex-A15 might falsely trigger a watchpoint exception on a CLREX instruction
52	New	832972	CatC		HSTR.{T7,T8,T15} bits incorrectly trap CDP instructions

16 May 2014: Changes in Document v14

Page	Status	ID	Cat	Rare	Summary of Erratum
30	New	827671	CatB		WriteClean and WriteBack transaction reordering might cause data corruption
37	New	827923	CatB	Rare	TLB maintenance operations might not be synchronized by DSB instruction
50	New	826375	CatC		Debug accesses in User mode do not properly generate undefined instruction exceptions for some SIMD and VFP registers
51	New	826969	CatC		Cortex-A15 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary

10 Mar 2014: Changes in Document v13

Page	Status	ID	Cat	Rare	Summary of Erratum
49	New	822719	CatC		Read following a write of a Timer TVAL register might return incorrect value

11 Dec 2013: Changes in Document v12

Page	Status	ID	Cat	Rare	Summary of Erratum
48	New	816469	CatC		Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption

22 Nov 2013: Changes in Document v11

Page	Status	ID	Cat	Rare	Summary of Erratum
29	New	816470	CatB		DSB instruction in processor power down sequence may not drain all required transactions

01 Nov 2013: Changes in Document v10

Page	Status	ID	Cat	Rare	Summary of Erratum
27	New	813469	CatB		An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

22 Sept 2013: Changes in Document v9

Page	Status	ID	Cat	Rare	Summary of Erratum
36	New	812169	CatB	Rare	DVM message blocked by copyback on AMBA Chi interconnect
46	New	808469	CatC		Deprecated SWP/SWPB can cause a stall until the next interrupt

21 Aug 2013: Changes in Document v8

Page	Status	ID	Cat	Rare	Summary of Erratum
47	New	810072	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned
44	Updated	803670	CatC		Unaligned load accessing two cache lines could return uncorrected data in case of single bit ECC error in the L2 cache

05 June 2013: Changes in Document v7

Page	Status	ID	Cat	Rare	Summary of Erratum
17	New	804623	CatA		Three ACE Snoops colliding with three readUniques might cause deadlock
26	New	804622	CatB		Extension register load to SO/Dev address combined with erroneous BLX can stall core
35	New	804969	CatB	Rare	Extension register loads and stores can livelock core
23	Updated	798181	CatB		Moving a virtual page that is being accessed by an active process can lead to unexpected behavior

24 Apr 2013: Changes in Document v6

Page	Status	ID	Cat	Rare	Summary of Erratum
25	New	803619	CatB		Clash of DCCMVAC with snoop could cause data corruption
43	New	803620	CatC		Near zero Advanced SIMD fused multiply add may be incorrectly flushed to zero
44	New	803670	CatC		Unaligned load accessing two cache lines could return uncorrected data in case of single bit ECC error in the L2 cache
45	New	803671	CatC		The ACE ARDOMAIN field may be incorrect on some Instruction or Tablewalk prefetch requests

21 Mar 2013: Changes in Document v5

Page	Status	ID	Cat	Rare	Summary of Erratum
18	New	801819	CatA	Rare	An eviction from L1 data cache might stall indefinitely in the L2 write buffer preventing a snoop from completing
23	Updated	798181	CatB		Moving a virtual page that is being accessed by an active process can lead to unexpected behavior

14 Feb 2013: Changes in Document v4

There are no New or Updated errata associated with this release.

31 Jan 2013: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
16	New	799271	CatA		A combination of Non-cacheable or streaming writes, a DSB instruction, and a snoop can cause a core to deadlock

23	New	798181	CatB	Rare	Moving a virtual page that is being accessed by an active process can lead to unexpected behavior
33	Updated	763126	CatB	Rare	Three processor exclusive access livelock

15 Nov 2012: Changes in Document v2

No new or updated errata. REVID[0] used to indicate an ECO Patch for Defect 794724.

23 Oct 2012: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
15	New	794724	CatA		L2 Cache initialization sequence may not function correctly if the L2 tag RAM requires two-cycle setup time
33	Updated	763126	CatB	Rare	Three processor exclusive access livelock

Copyright © 2015 ARM. All rights reserved.

Non Confidential

Contents

Chapter 1. **Introduction**

This chapter introduces the errata notice for the ARM Cortex-A15 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.

This document includes errata that affect revision r3 only.

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

Table 2 Revisions Affected

ID	Cat	Rare	Summary of Erratum	r3p0	r3p1	r3p2	r3p3
784421	CatA		CPU may deadlock if short interrupt pulse deasserts just as a WFI instruction is executed	X			
788420	CatA		A15 can stall if 16 stores are issued and the 16th store gets its BRESP before any of the earlier stores get their BRESP	X	X		
794724	CatA		L2 Cache initialization sequence may not function correctly if the L2 tag RAM requires two-cycle setup time	X	X	X	
799271	CatA		A combination of Non-cacheable or streaming writes, a DSB instruction, and a snoop can cause a core to deadlock	X	X	X	
804623	CatA		Three ACE Snoops colliding with three readUniques might cause deadlock	X	X	X	X
801819	CatA	Rare	An eviction from L1 data cache might stall indefinitely in the L2 write buffer preventing a snoop from completing	X	X	X	
784420	CatB		Speculative instruction fetches with MMU disabled might not comply with architectural requirements	X	X	X	X
784477	CatB		CTIINTACK register needs clearing each time it is set	X	X	X	
785769	CatB		Undefined exception is not generated for LDC/STC instructions which access DCC registers in User mode	X	X	X	X
798181	CatB		Moving a virtual page that is being accessed by an active process can lead to unexpected behavior	X	X	X	
803619	CatB		Clash of DCCMVAC with snoop could cause data corruption	X	X	X	X
804622	CatB		Extension register load to SO/Dev address combined with erroneous BLX can stall core	X	X	X	X

ID	Cat	Rare	Summary of Erratum	r3p0	r3p1	r3p2	r3p3
813469	CatB		An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall	X	X	X	X
816470	CatB		DSB instruction in processor power down sequence may not drain all required transactions	X	X	X	X
827671	CatB		WriteClean and WriteBack transaction reordering might cause data corruption	X	X	X	X
830321	CatB		Cortex-A15 might falsely trigger a watchpoint exception on a CLREX instruction	X	X	X	
836969	CatB		Code sequence continuously hitting the L1 cache can block snoop	X	X	X	X
763126	CatB	Rare	Three processor exclusive access livelock	X	X	X	X
804969	CatB	Rare	Extension register loads and stores can livelock core	X	X	X	X
812169	CatB	Rare	DVM message blocked by copyback on AMBA Chi interconnect	X	X	X	X
827923	CatB	Rare	TLB maintenance operations might not be synchronized by DSB instruction	X	X	X	X
773023	CatC		Order may not be maintained between Strongly Ordered memory requests and Device memory requests on ACE/AXI	X	X	X	X
777769	CatC		ICache parity error may not be corrected for NC code	X	X	X	X
784419	CatC		An unaligned load crossing a 4K boundary between SO/Dev memory and WB memory can stall the core until the next interrupt	X	X	X	
784469	CatC		CTI Authentication Status register is incorrect	X	X	X	
788419	CatC		If a single memory address is aliased as both shareable and non-shareable, a CMO to this same line might deadlock	X	X	X	
803620	CatC		Near zero Advanced SIMD fused multiply add may be incorrectly flushed to zero	X	X	X	X
803670	CatC		Unaligned load accessing two cache lines could return uncorrected data in case of single bit ECC error in the L2 cache	X	X	X	X
803671	CatC		The ACE ARDOMAIN field may be incorrect on some Instruction or Tablewalk prefetch requests	X	X	X	X
808469	CatC		Deprecated SWP/SWPB can cause a stall until the next interrupt	X	X	X	
810072	CatC		When a single-bit ECC error occurs in the L2, uncorrected data might be returned	X	X	X	X
816469	CatC		Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption	X	X	X	X
822719	CatC		Read following a write of a Timer TVAL register might return incorrect value	X	X	X	X
826375	CatC		Debug accesses in User mode do not properly generate undefined instruction exceptions for some SIMD and VFP registers	X	X	X	X
826969	CatC		Cortex-A15 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary	X	X	X	X
832972	CatC		HSTR.{T7,T8,T15} bits incorrectly trap CDP instructions	X	X	X	X
834569	CatC		PMU event BUS_CYCLES might be incorrect in some cases	X	X	X	X
842119	CatC		Instruction issued through ITR by debugger executes incorrectly for PCLKDBG frequencies much slower than the processor	X	X	X	X
851024	CatC		Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores	X	X	X	X

r3p2 implementation fix

Note the following errata may be fixed in some implementations of r3p2. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[0]	794724 L2 Cache initialization sequence may not function correctly if the L2 tag RAM requires

	two-cycle setup time
REVIDR[1]	799271 A combination of Non-cacheable or streaming writes, a DSB instruction, and a snoop can cause a core to deadlock
REVIDR[2]	Not used in product revision r3pX
REVIDR[3]	801819 An eviction from L1 data cache might stall indefinitely in the L2 write buffer preventing a snoop from completing
REVIDR[4]	798181 Moving a virtual page that is being accessed by an active process can lead to unexpected behavior
REVIDR[5]	804623 Three ACE Snoops colliding with three readUniques might cause deadlock
REVIDR[6]	804622 Extension register load to SO/Dev address combined with erroneous BLX can stall core
REVIDR[7]	804969 Extension register loads and stores can livelock core
REVIDR[8]	Reserved
REVIDR[9]	798181 Moving a virtual page that is being accessed by an active process can lead to unexpected behavior – This is an update to a previously released ECO.
REVIDR[10]	803670 Unaligned load accessing two cache lines could return uncorrected data in case of single bit ECC error in the L2 cache 810072 When a single-bit ECC error occurs in the L2, uncorrected data might be returned
REVIDR[11]	Reserved
REVIDR[12]	813469 An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

Note that there is no change to the MIDR which remains at r3p2, but the REVIDR might be updated from 0x00 to indicate that one or more errata are corrected as shown above. Software will identify this release through the combination of MIDR and REVIDR.

r3p3 implementation fix

Note the following errata may be fixed in some implementations of r3p3. This can be determined by reading the REVIDR register where a set bit indicates that the erratum is fixed in this part.

REVIDR[0]	Reserved
REVIDR[1]	Reserved
REVIDR[2]	Not used in product revision r3pX
REVIDR[3]	Reserved
REVIDR[4]	Reserved
REVIDR[5]	804623 Three ACE Snoops colliding with three readUniques might cause deadlock
REVIDR[6]	804622 Extension register load to SO/Dev address combined with erroneous BLX can stall core
REVIDR[7]	804969 Extension register loads and stores can livelock core
REVIDR[8]	Reserved
REVIDR[9]	798181 Moving a virtual page that is being accessed by an active process can lead to unexpected behavior – This is an update to a previously released ECO.
REVIDR[10]	803670 Unaligned load accessing two cache lines could return uncorrected data in case of single bit ECC error in the L2 cache 810072 When a single-bit ECC error occurs in the L2, uncorrected data might be returned

REVIDR[11]	Reserved
REVIDR[12]	813469 An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

Note that there is no change to the MIDR which remains at r3p3, but the REVIDR might be updated from 0x00 to indicate that one or more errata are corrected as shown above. Software will identify this release through the combination of MIDR and REVIDR.

2.3. Category A

784421: CPU may deadlock if short interrupt pulse deasserts just as a WFI instruction is executed

Category A

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0. Fixed in r3p1.

Description

If an interrupt is asserted to a CPU and then deasserted before that CPU takes the interrupt and that CPU is executing a WFI instruction at the same time, it is possible that the CPU will enter WFI state and never wake up.

Note: This erratum matches bug #5362 in the ARM internal Jira database.

Conditions

The erratum can occur only if the following sequence of conditions is met:

1) The CPU starts to execute a WFI instruction, with no interrupt pending.
2) An interrupt is asserted. This can be a physical or a virtual interrupt, from an interrupt pin or from the internal GIC.
3) The Interrupt is deasserted without being taken by the CPU.
4) The CPU completes execution of the WFI instruction and enters WFI mode.

Implications

If the above conditions occur, it is possible that the CPU will never wake up from WFI mode. Subsequent interrupts to the CPU will be ignored.

Workaround

There is no useful workaround.

788420: A15 can stall if 16 stores are issued and the 16th store gets its BRESP before any of the earlier stores get their BRESP

Category A

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1. Fixed in r3p2.

Description

A15 issues 15 writes on the AXI write channel without receiving a BRESP for any of them. A15 then issues a 16th write and receives a BRESP for the 16th write before receiving a BRESP for any of the first 15 writes. If this occurs, A15 might deadlock.

Note: This erratum matches bug #5380 in the ARM internal Jira database.

Conditions

1) A15 issues 16 stores on the AXI write channel without receiving a BRESP.
2) A BRESP is returned for the 16th store before a BRESP is received for any of the earlier stores.

Implications

If the above conditions occur A15 might deadlock.

Workaround

There is no software workaround.

794724: L2 Cache initialization sequence may not function correctly if the L2 tag RAM requires two-cycle setup time

Category A

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

If a Cortex-A15 MPCore implementation uses an L2 tag RAM that requires a two cycle setup time for address and data, the L2 cache might not be correctly invalidated by the hardware initialization sequence that occurs after the deassertion of nL2RESET.

By default, the Cortex-A15 MPCore L2 tag RAM input paths are single cycle paths for both setup and hold. However, if the L2 tag RAM used in an implementation requires it, software can program L2CTLR[9] to 1 to configure the setup paths to be two cycle multicycle paths. These must be set correctly before the data cache is enabled.

Cortex-A15 MPCore initializes the L2 cache in hardware when the L2 is reset. Because this hardware initialization occurs before the L2CTLR register can be programmed by software, the hardware sequence must assume tag RAM array timings that will work with all legal RAM instances.

During hardware RAM initialization of the L2 on affected versions of Cortex-A15 MPCore the setup used for the L2 tag RAM is a single cycle. It should be two cycles to support RAMs that require more setup.

Note: This erratum matches bug #5403 in the ARM internal Jira database.

Configurations Affected

This erratum affects implementations that require L2CTLR[9] to be set to 1 because the L2 tag RAM requires two cycle setup.

This erratum does not affect the processor if the revision and variant reported by the MIDR is r3p0, r3p1 or r3p2 and the REVIDR[0] is set to 1.

Conditions

This erratum requires the following condition:

- The implementation uses an L2 tag RAM that requires two cycle setup on address and data. Affected implementations will program L2CTLR[9]=1 before enabling the data cache.

Implications

The L2 Cache may not be correctly initialized after deassertion of nL2RESET.

1) Valid lines with unknown addresses and data might be present in the cache after reset. If the valid lines match any address in physical memory that is accessed before the lines are evicted, they could deliver incorrect data on reads.
2) If the lines appear valid and dirty, they could generate spurious memory transactions that would corrupt memory or generate errors in the system.

Workaround

Using a lower frequency such that the tag RAM instance can meet timing without the two cycle setup is a valid system workaround. The lower frequency need only be used between deassertion of nL2RESET and the completion of the cache initialization sequence. Software can ensure that the hardware initialization sequence has completed by executing any data cache maintenance operation (e.g. DCCISW) followed by a DSB. The maintenance operation will not complete until the L2 initialization is complete.

Software initialization of the L2 cache is not a valid workaround. The DCISW invalidate by set/way instruction is treated by A15 as a DCCISW clean/invalidate by set/way instruction. At the end of a software initialization of the L2 cache, the cache would be correctly invalid, but the software initialization could cause spurious evictions of lines incorrectly marked as valid and dirty.

799271: A combination of Non-cacheable or streaming writes, a DSB instruction, and a snoop can cause a core to deadlock

Category A

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

If a certain combination of snoops, DSB instruction execution, and stores occurs with a particular timing, a Cortex-A15 MPCore CPU can deadlock. The deadlocked CPU will be unable to complete an eviction until earlier writes complete, and those writes will not be able to complete until that eviction completes.

Note: This erratum matches bug #5416 in the ARM internal Jira database.

Configurations Affected

This erratum does not affect the processor if the revision and variant reported by the MIDR is r3p0, r3p1 or r3p2 and REVIDR[1] is set to 1.

Conditions

1) One core (coreX) is executing store instructions that are not allocating to the L1 cache. These will be a combination of:
   - Writes to Non-cacheable, Strongly-ordered, or Device memory locations.
   - Write-streaming full line cacheable writes.
2) These stores back up in the memory system and allocate 12 entries of the L2 Write Request Queue
3) CoreX begins the eviction of a cache line (A) that is being replaced by a returning cache line fill
4) Another core (coreY) executes a DSB instruction that generates a TLB synchronization request to coreX
5) A memory request occurs (from coreZ, an ACP request, an external snoop, or a translation table walk request) that
   - hits the cache line (A) being evicted from coreX, OR
   - hits another line in the L1 data cache of coreX that is in uniqueClean or uniqueDirty state.

Implications

If the above scenario occurs with a particular timing, it is possible that the eviction from coreX cannot complete until another L2 write buffer credit is returned, and no such credit can be returned until the eviction completes. In this situation, the processor deadlocks.

In a single core configuration without ACE, the erratum cannot not occur.

In a single core configuration with ACE, but with no other ARM cores capable of issuing a “DVM Sync” request, the erratum cannot occur.

In a two core configuration without ACE, the erratum is believed to be very rare.

In a two core configuration with ACE, or a three or more core configuration, the erratum can occur.

Workaround

There is no known workaround.

804623: Three ACE Snoops colliding with three readUniques might cause deadlock

Category A

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

Three cacheable memory requests are issued that hit the L2 cache in shared state and require unique state. Three readUnique requests are issued for the three cache lines (A, B, C). Before any response occurs for the readUnique requests, three snoops are received on the AC snoop channel for the three cache lines. These snoops look up the cache and generate responses on the CR and CD channels. All three snoops are then scheduled to update the cache state of their respective lines. Before these cache update requests are processed, the readUnique read data responses are returned for all three cache lines. The three readUnique requests are scheduled to update the cache state of the three lines.

The readUnique cache updates will be issued first, but will hazard against the snoops to the same line. The readUnique cache updates should then stop issuing until the snoops cache updates complete. However, if this erratum occurs, the three readUnique requests will continue to issue and hazard, preventing forward progress for the snoops.

Note: This erratum matches bug #5447 in the ARM internal Jira database.

Configurations Affected

ACE interconnect only

Conditions

1) Three readUnique issues on ACE for cache lines (A, B, C) that are in shared state in the L2 cache.
2) AC channel snoops are received for each of the three cache lines.
3) The cache updates associated with the snoops are held off for many cycles.
4) The readUnique requests receive their responses and data from ACE before the snoops update the cache.

Implications

If the erratum conditions are met, the L2 cache might deadlock.

Because the cache updates scheduled by the snoops are generally processed very quickly (5-10 cycles), and readUnique responses from the bus will typically require 50+ cycles to complete the required snoops or to read the data from memory, this erratum is not expected to be common.

Workaround

There is no workaround..

2.4. Category A (Rare)

801819: An eviction from L1 data cache might stall indefinitely in the L2 write buffer preventing a snoop from completing

Category A Rare

Products Affected: Cortex-A15 MP Core -NEON.
Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

A livelock can occur in the L2 cache arbitration that might prevent a snoop from completing. Under certain conditions this can cause the system to deadlock.

When a cacheable store is executed in the L2 it requires an entry to be allocated in the Fill Evict Queue (FEQ). If an entry is not available the store will restart, as will any younger stores from the same CPU, because stores from the same CPU must execute in order.

If two cacheable stores from the same CPU issue into the L2 just as an FEQ entry becomes available, it is possible for the first store (ST1) to detect the FEQ as full, but the second store (ST2) to see an available FEQ entry and allocate that entry. Both stores are then restarted, the ST1 due to the lack of an FEQ entry, ST2 because the ST1 restarted. This means the FEQ entry temporarily allocated by ST2 is marked for deallocation.

The two stores will then reissue to the L2. If the timing of the reissue aligns with the deallocation of the FEQ entry temporarily allocated previously by ST2, it is possible that ST1 will again detect an FEQ full condition and ST2 will see an FEQ entry available and allocate it. If there is no other activity in the L2 to change the timings, this cycle can repeat until a second FEQ entry becomes available and both stores can proceed, or another FEQ entry is allocated for an unrelated request and both stores stop reissuing waiting for an FEQ entry to become available.

Depending on the L2 cache configuration options there might need to be one or more unrelated stores between ST1 and ST2 in order to hit the required timing for the livelock.

If there is an eviction in the write buffer that is behind the stores then that eviction might be stalled as long as the livelock continues. This stalled eviction might stall a snoop hit on the L1 data cache of that CPU. The stalled snoop might block further FEQ entries from deallocating, either through a dependency in the ACE memory system, or because the FEQ is full of snoop hits to the L1 data cache of that CPU. If all of these conditions occur the system can deadlock.

Note: This erratum matches bug #5433 in the ARM internal Jira database.

Configurations Affected

This erratum does not affect the processor if the revision and variant reported by the MIDR is r3p0, r3p1, or r3p2 and REVIDR[3] is set to 1.

Conditions

1) The L2 Fill Evict Queue (FEQ) is full.
2) Cacheable write ST1 issues followed by cacheable write ST2 from the same CPU.
3) An eviction is waiting in the write buffer behind ST1, ST2, and one or more additional stores.
4) A snoop (from another A15 core or from the ACE AC channel) is waiting for that eviction to complete.
5) That snoop is preventing further FEQ entries from deallocating.
6) Very specific timing conditions.

Implications

If the erratum conditions are met the part will deadlock.

Cortex-A15 will only issue two types of cacheable stores from a CPU into the L2: Write-Back No-Allocate stores, or Write-Back stores that have been gathered into a full line write using the L1 write streaming logic.

For implementations of Cortex-A15 configured with the “L2 arbitration register slice” (arb-slice) option (typically four core systems) the only cacheable stores that can cause the issue are Write-Back No-Allocate stores. Because Write-Back No-Allocate stores are not commonly used and are easy to avoid, there is a straightforward workaround for these implementations. For configurations with the arb-slice, this erratum is Category B.

For implementations of Cortex-A15 configured without the arb-slice option (typically 1 or 2 core systems) Write-Back No-Allocate stores and streaming cache line writes can both lead to the deadlock, although hitting the conditions with streaming cache line writes is much more difficult. There is still a full workaround, but because it involves disabling write-streaming there is a performance hit on some streaming memory workloads. For configurations without the arb-slice, this erratum is Category A Rare.

Workaround

Do both of the following:

1) Do not use the write-back no-allocate memory type.
2) Do not issue write-back cacheable stores at any time when the cache is disabled (SCTLR.C=0) and the MMU is enabled (SCTLR.M=1). Because it is implementation defined whether cacheable stores update the cache when the cache is disabled it is not expected that any portable code will execute cacheable stores when the cache is disabled.

For implementations of Cortex-A15 configured without the “L2 arbitration register slice” option (typically one or two core systems), you must also do the following:

3) Disable write-streaming in each CPU by setting ACTLR[28:25] = 0b1111

2.5. Category B

784420: Speculative instruction fetches with MMU disabled might not comply with architectural requirements

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3

Description

When all applicable stages of translation are disabled, an ARMv7 processor must follow some architectural rules regarding speculative fetches and the addresses to which these can be initiated. These rules avoid potential reads to read-sensitive areas. For more information about these rules see the description of "Behavior of instruction fetches when all associated MMUs are disabled" in the ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition. Cortex-A15 normally operates with both the MMU and branch prediction enabled. If the processor operates in this condition for any significant amount of time, the BTB (branch target buffer) will contain branch predictions. If both stages of translation are then disabled, but branch prediction remains enabled, these stale BTB entries can cause A15 to violate the rules for speculative fetches.

Note: This erratum matches bug #5360 in the ARM internal Jira database.

Conditions

The erratum can occur only if the following sequence of conditions is met:

1) MMU enabled for at least one stage of address translation

2) Branch prediction enabled

3) Branches executed

4) MMU disabled for all applicable stages of address translation

Note: When executing in a Non-secure PL1 or PL0 mode, for condition 1 at least one stage of address translation must be enabled, and for condition 4 both stages of address translation must be disabled. When executing in any other mode, there is only one stage of address translation, that must be enabled for condition 1 and disabled for condition 4.

Implications

If the above conditions occur, it is possible that after the MMU is disabled, speculative instruction fetches will occur to read-sensitive locations.

Workaround

Branch prediction should be disabled when the MMU is disabled after having been enabled. This should be done by clearing the appropriate Z bit in the System Control register at the same time as or just before the final stage of translation is disabled. Branch prediction should remain disabled until the MMU is enabled, or until the BTB has been flushed. On A15, the BPI* branch predictor maintenance commands will not invalidate the BTB. The BTB can be flushed by setting bit 0 of the ACTLR register, doing any instruction cache invalidate instruction (e.g. ICIALLU), and then clearing bit 0 of the ACTLR register.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

784477: CTIINTACK register needs clearing each time it is set

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

The CTI contains a CTIINTACK register, which enables a trigger to be acknowledged through software, instead of using a hardware knowledge using the CTITRIGOUTACK input. The correct operation of this register is that writing a one to the bit corresponding to a trigger output will cause that trigger to be cleared, and this will not affect future triggers.

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

The CTIINTACK register is normally used in two cases:

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

Conditions

The following conditions must occur:

- A CTI trigger output fires.
- The CTI CTIINTACK register is used to acknowledge the trigger output, by writing a one to the bit corresponding to that trigger output.
- The same trigger output fires again before the corresponding bit in the CTIINTACK register is cleared.

Implications

Trigger outputs might be missed:

- In the case of a debug-originated interrupt that uses CTIINTACK to clear the interrupt, events other than the first event might not cause an interrupt to occur.
- In the case of a cross-halting debug request, after the first time a processor halts and restarts, it might halt without halting other processors with it.

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 occurring between the two register writes might be lost. This is in general not 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.

785769: Undefined exception is not generated for LDC/STC instructions which access DCC registers in User mode

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3

Description

The Debug Communication Channel (DCC) registers are accessible using MCR/MRC instructions. LDC/STC instructions provide alternate access to DCC registers DBGDTRTXint/DBGDTRRXint.

In Non-debug state when DBGDSCR.UDCCdis is set to 1, then access to DCC register using MCR/MRC and LDC/STC instructions from User mode should generate an Undefined Instruction exception. Access using MCR/MRC instructions correctly generates an Undefined Instruction exception correctly. However access using LDC/STC instructions does not generate the Undefined Instruction exception and incorrectly accesses the register.

Note: This erratum matches bug #5372 in the ARM internal Jira database.

Conditions

1) The processor is in Non-debug state and User mode.
2) DBGDSCR.UDCCdis is set to 1.
3) Either the Hypervisor trap to debug registers in not set (HDCR.TDA==0) or the processor is in Secure state.
4) LDC to DBGDTRTXint or STC to DBGDTRRXint is executed.

Implications

If LDC or STC instructions are executed in User mode, then DCC traffic between debug host and debug target from FIQ/IRQ/Supervisor/Monitor/Abort/Hypervisor/Undefined/System modes can be corrupted due to this errata.

Workaround

Tools must use MCR/MRC instructions to access Debug Communication Channel (DCC) registers from User mode, instead of LDC/STC instructions, in Non-debug state.

Alternatively, software can avoid corruption of DCC traffic by Non-secure User mode code by setting the hypervisor trap for debug register accesses (HDCR.TDA), and handling the LDC/STC instruction appropriately in hypervisor code. There is no software workaround to prevent LDC/STC instructions executing in Secure User mode from corrupting DCC traffic handled in other processor modes.

798181:Moving a virtual page that is being accessed by an active process can lead to unexpected behavior

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

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-A15 this might not occur in the following cases:

- The exclusive monitor is physical address based. Moving a memory region in physical memory might cause unexpected results, including multiple threads acquiring the same lock.
- Hazarding logic to guarantee read after read ordering to the same address is physical address based. If a memory region is moved in physical memory ordering violations might occur.
- The DSB instruction might complete before all memory transactions to the invalidated translation are globally observed.

The first two conditions might lead to unexpected ordering and Load-Exclusive/Store-Exclusive instructions behaving in an unexpected way. The third condition 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 memory transactions have completed.

Note: This erratum matches bugs #5405, #5407, #5428, #5429 in the ARM internal Jira database.

Configurations Affected

This erratum does not affect the processor if the revision and variant reported by the MIDR is r3p0, r3p1 or r3p2 and REVIDR[4] is set to 1.

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:

1) exclusive monitors in the system might be using the old translation
   - this might lead to unexpected Load-Exclusive/Store-Exclusive behavior.
2) the read after read hazard logic might be using the old translation
   - this might lead to a load instruction returning older data than a previous load instruction that accessed the same virtual memory location.
3) outstanding loads or 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

To ensure the completion of a TLB invalidate, software must first follow the translation table invalidation steps specified in the ARM Architecture Reference Manual:

1) Modify the translation tables in memory as required.

2) Execute a DSB to ensure observation of the stores to the translation tables.

3) Execute one or more TLB invalidate instructions targeting the affected memory regions.

4) Execute a DSB to ensure the TLB invalidate instructions from step 3 have been propagated to all processors in the system.

Next, on the processor that is performing the translation table maintenance, it must also perform these two additional steps:

5) Execute a dummy TLBIMVAIS, meaning a TLBIMVAIS to any address with any ASID.

6) Execute a DSB instruction to ensure the dummy TLB invalidate instruction has been propagated to all processors in the system.

Finally, identify any Cortex-A15 processors in the system that are currently using the translation context of the region being invalidated. For global regions this will be all Cortex-A15 processors not powered down or in reset. For non-global regions this will be all Cortex-A15 processors that are currently using the same translation context (typically ASID and VMID) as the modified translation table entries. On each of those processors, ensure that the following occur (in any order):

- execution of a CLREX instruction
- execution of a DMB or DSB instruction

Once the above sequence is complete all uses of the old translation in the system will removed, and any memory transactions to the old translation will be globally observed.

To meet the final requirement, the processor performing the translation table maintenance must execute a CLREX and a DMB or DSB instruction.

The easiest way to meet the final requirement on other processors in the system is to send an inter-processor interrupt to each required processor. Most interrupt handlers will execute a CLREX instruction (as required by the ARM Architecture on a context switch). The interrupt handler must execute a DMB instruction and then signal completion to the processor doing the translation table maintenance.

Note: this workaround must be implemented on any ARM Architecture processor that is executing TLB invalidate instructions that may affect a Cortex-A15 processor. This would include translation table maintenance being performed on a Cortex-A7 processor when a Cortex-A15 processor is present in the system.

In r3p2 and earlier versions with REVIDR[4]= 0, the full workaround is required.

In r3p2 and earlier versions with REVIDR[4]=1, REVIDR[9]=0, only the portion of the workaround up to the end of step 6 is required.

In r3p2 and earlier versions with REVIDR[4]=1, REVIDR[9]=1, no workaround is required.

In r3p3, if REVIDR[9]=0, only the portion of the workaround up to the end of step 6 is required.

In r3p3, if REVIDR[9]=1, no workaround is required..

Copyright © 2015 ARM. All rights reserved.

Non Confidential

803619: Clash of DCCMVAC with snoop could cause data corruption

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

If a DCCMVAC cache clean operation collides with a non-invalidating ACE snoop request to the same cache line, in rare cases the copy of the line retained in the L2 cache can be corrupted.

Note: This erratum matches bug #5437 in the ARM internal Jira database.

Configurations Affected

ACE systems only

Conditions

1) A DCCMVAC instruction is executed to clean cache line A.
2) Cache line A is held in unique state in an L1 data cache.
3) A non-invalidating snoop is received on the AC channel targeting cache line A.
4) The external snoop is a ReadOnce, ReadClean, ReadShared, ReadNotSharedDirty or CleanShared.
5) The snoop is processed after the line has been cleaned from the L1 by the DCCMVAC but before the corresponding ACE cache maintenance operation has been issued on the AR channel.
6) The response to the ACE cache maintenance operation is received very quickly, before the L2 cache is updated with the latest data from the L1 cache.
7) The ACE cache maintenance operation response corrupts the data in the buffer, and this corrupted data is then written to the L2 cache.

Implications

If the erratum conditions are hit, data in the L2 cache is corrupted.

Workaround

This erratum can be avoided by setting ACTLR2[0] = 1. This bit turns all DCCMVAC and DCCSW operations into the corresponding clean and invalidate operations (DCCIMVAC and DCCISW) which are not vulnerable to this erratum.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

804622: Extension register load to SO/Dev address combined with erroneous BLX can stall core

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

An extension register load instruction (ExLd1) is executed at instruction address Z. The memory location being loaded by ExLd1 is in Strongly Ordered or Device (SO/Dev) memory. This extension register load to SO or Dev memory triggers a special mechanism that includes a pipe flush and re-execution of ExLd1 in a special ordering mode.

Before the ExLd1 pipe flush occurs, two branch instructions (B1 and B2) are issued. B1 either branches back to Z but incorrectly switches to Thumb state (if currently in ARM state), or branches to a misaligned address near Z (if currently in Thumb state). This misalignment or incorrect instruction set state causes the two bytes at Z to be decoded as a branch (B2) instead of as a load. This bad branch B2 is speculatively executed and loads the branch target buffer (BTB) with information indicating instruction address Z is a branch.

The ExLd1 pipe flush then occurs, and address Z is refetched. The BTB now incorrectly predicts address Z as a branch. When the incorrect prediction is detected the instruction is discarded and the pipe flushed. However, this bad-branch pipe flush exits the special ordering mode needed by ExLd1. When Z is refetched, the attempted re-execution of ExLd1 again triggers the SO/Dev pipe flush and the sequence repeats.

An interrupt or other exception permits normal execution to resume in the exception handler and other threads. However, upon return to the original thread the same repeated flushes might occur.

Note: This erratum matches bug #5449 in the ARM internal Jira database.

Conditions

1) An extension register load instruction (ExLd1) at address Z to Device or Strongly Ordered memory
2) An erroneous branch, either:
    - In ARM instruction set state a branch targeting address Z, but incorrectly switching to Thumb instruction set state, OR
    - In Thumb instruction set state, a branch targeting a 32 bit instruction at (Z-2) overlapping the ExLd1.

Implications

If the conditions are met, the core will stop executing instructions in the current thread. An interrupt or other external exception will break the stall, but upon returning from the interrupt the same sequence might deadlock again. The effect would be that the affected thread is never able to make forward progress.

This erratum is not expected to be common. Extension register loads to Strongly Ordered or Device memory are rare in normal applications, and the 'bad branch' should not occur in executable code as the results would be unpredictable. However, if an extension register load was followed by an unpredictable branch over a block of data it is possible that early bytes in the block of data could be decoded as the bad branch back to the extension register load.

Workaround

If a thread is unable to make forward progress and the next instruction to be executed is an extension register load to Strongly Ordered or Device memory that meets the erratum requirements, ensure that there is a predictable branch between the load and the potential bad branch. If there is a data section soon after an extension register load to SO/Dev memory where the data might be decoded as a branch, with no intervening predictable branch, placing a dummy "branch to self" at the head of the data section would avoid the erratum.

813469: An unaligned store instruction crossing a 4k page boundary at the same time as the lower page is invalidated might stall

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

In a very unusual timing boundary condition, an unaligned store instruction crossing a 4k boundary at the same time as the lower page is being invalidated might stall until the next interrupt. If the store instruction is a VSTM of 88 bytes or larger, not 8-byte aligned, with 80 of the bytes in the lower page, the stall will not be broken by an interrupt and the core will stall until the next reset.

Note: This erratum matches bug #5469 in the ARM internal Jira database.

Conditions

1) A store is executed that crosses a 4k boundary.
2) The store has the following alignment:
   - 2-byte store, not 2-byte aligned
   - 4-byte store, not 4-byte aligned
   - 8-byte or larger store, not 8-byte aligned
3) The lower page hits a valid translation in the L1DTLB.
4) The upper page misses in the L1DTLB and the table walk returns a translation fault.
5) While the TLB miss for the upper page is being serviced, the lower page translation is invalidated by a TLBI instruction from an ARM core.
6) The lower page misses in the L1DTLB and the table walk returns a translation fault.

- All of the above conditions are required to hit the basic erratum, which will generally be broken by the next interrupt. To hit the stall that will not be broken by an interrupt, further conditions are required:

7) The store is a VSTM of 88 or more bytes, 4-byte aligned but not 8-byte aligned.
8) The first 80 bytes of the VSTM are in the lower page, with some of the bytes in the upper page.

Implications

If conditions 1-6 above occur with the correct timing, the CPU will stall until the next interrupt. If conditions 1-8 occur, that CPU will stall indefinitely, broken only by resetting that CPU.

The erratum will only occur in rare boundary conditions when an OS is invalidating a page that is actively being used by a process on another CPU and all the conditions are met. Because hitting the erratum is expected to be quite rare, in a system where the A15 CPUs get periodic interrupts the impact in most systems is likely to be minimal.

Compilers are not expected to generate an unaligned VSTM large enough to cause the indefinite stall. Optimized memcpy using Neon registers uses VSTR, not VSTM. Function prologues may use VSTM, but are not expected to store more than 64 bytes with normal calling conventions, even if the stack were not 8 byte aligned. Large VSTMs might occur in cases such as exception unwinding and state saving, but in those cases the VSTM will be 8-byte aligned and not affected. The VSTM indefinite stall case is therefore not expected to occur in a typical system.

Workaround

The suggested workaround is to avoid the use of large unaligned VSTM and to supply periodic interrupts to each CPU to clear the stall should it ever occur.

If large unaligned VSTM cannot be avoided, interrupts are not available, or a rare delay until the next interrupt is not acceptable, an alternative workaround is to do local TLB maintenance on each CPU rather than using the distributed TLB maintenance mechanism:

- Do TLB maintenance locally on each CPU using the local (not “inner shared”) TLB maintenance operations.

- Between TLB maintenance operations and the subsequent required DSB instruction, do not execute unaligned page crossing stores to the pages being invalidated.

816470: DSB instruction in processor power down sequence may not drain all required transactions

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

When powering down an individual core in a Cortex-A15 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 section 2.4.3 of the A15 TRM, the ACTLR.SMP 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. On A15 r2 and earlier this sequence works correctly. However, on the r3 and later versions of A15, 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 #5472 in the ARM internal Jira database.

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

827671: WriteClean and WriteBack transaction reordering might cause data corruption

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

A cache maintenance operation (DCCMVAC or DCCSW instruction) that targets a dirty line in the L2 cache will generate a WriteClean transaction on the ACE interface. If a subsequent transaction causes a replacement of the line from the L2 cache then a WriteBack could be generated before the WriteClean operation completed. If the interconnect allows re-ordering between the WriteClean and WriteBack operations, data corruption might occur.

Note: This erratum matches bug #5492 in the ARM internal JIRA database.

Configurations affected

Systems supporting:

- Interconnect supporting re-ordering of WriteClean and WriteBack transactions
- BROADCASTCACHEMAINT=0
- BROADCASTCACHEMAINT=1 and DCCSW instruction used to clean lines from Cortex-A15

Conditions

1) Processor executes DCCMVAC or DCCSW to dirty line in Cortex-A15 cluster triggering WriteClean transaction on ACE write channel.
2) Processor write marks line dirty.
3) Subsequent linefill triggers WriteBack of line prior to WriteClean transaction completing.
4) WriteBack and WriteClean transactions are reordered causing memory data corruption.

Note: the CCI-400 and CCN-504 interconnects, and DMC-400 memory controller will not reorder the WriteClean and WriteBack transactions, hence will not be exhibit the reordering behavior necessary for this erratum to occur.

Implications

If the above conditions exist, then the contents of the memory will be corrupted because the WriteClean and WriteBack requests are presented to the slave in a different order than issued on the ACE write channel.

Workaround

This erratum can be avoided by setting ACTLR2[0] = 1. This bit turns all DCCMVAC and DCCSW operations into the corresponding clean and invalidate operations (DCCIMVAC and DCCISW) which are not vulnerable to this erratum.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

830321: Cortex-A15 might falsely trigger a watchpoint exception on a CLREX instruction

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

Cortex-A15 might falsely trigger a watchpoint exception on a CLREX instruction.

Note: This erratum matches bug #5402 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

1) The watchpoint is set to match unprivileged instruction.
2) CLREX is executed. The irrelevant virtual address info from a previous execution happens to match the virtual address as being programmed in Watchpoint Value Register.

Note that this erratum will not occur if the watchpoint is not configured to trigger in the security state in which CLREX being executed. Also, this erratum will not occur if a CLREX is executed in Hyp mode (PL2) and the watchpoint is not configured to trigger in Hyp mode.

Implications

Cortex-A15 might falsely trigger a watchpoint exception on a CLREX instruction.

Workaround

- The watchpoint handler should look at the instruction that triggered the watchpoint. If it is CLREX, it should simply return from the handler.
- If user mode watchpoints are enabled, the kernel should not execute a CLREX in any situation where an abort cannot be tolerated.
- If a CLREX is required in a code sequence that cannot tolerate a precise abort, the CLREX should be replaced with a dummy STREX or user mode watchpoints should be disabled.

836969: Code sequence continuously hitting the L1 cache can block snoop

Category B

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3, r3p4.

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 #5500 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

1) A 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 Cortex-A15 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 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.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

2.6. Category B (Rare)

763126: Three processor exclusive access livelock

Category B Rare
Products Affected: Cortex-A15 MP Core -NEON.
Present in: r3p0, r3p1, r3p2, r3p3.

Description

In a system with three or more coherent masters that all use the ldrex/strex synchronization primitives to access a semaphore in coherent cacheable memory, there is a possibility of a livelock condition where two masters continuously attempt and fail to get the lock while the third master continuously reads the lock.

This erratum is heavily dependent on a unique set of initial conditions, and upon specific interconnect timing once the livelock has started. It is expected to be rare in a real system that the timing conditions will be hit.

An example: two cores C1 and C2 are contending for a lock using ldrex/strex, and core C3 is looping reading the same semaphore location. Once the livelock condition has started, from the perspective of C1, the sequence will look like this:

1) Execute ldrex, hits the cache in unique state.
2) External snoop takes line to shared state (triggered by C3 read).
3) Execute instructions to process the ldrex result and prepare the strex data.
4) Execute strex, hits cache shared, issues readUnique to bring in line unique.
5) External snoop invalidates line, clearing monitor (triggered by C2 strex that will eventually fail the monitor).
6) Line returns in unique state, but strex fails due to cleared monitor.
7) Loop back to step 1.

C1 and C2 constantly issue ReadUniques due to failing store exclusives that invalidate the line in the other core, each core causing the others strex to fail without making forward progress. No forward progress is made until/unless one of the cores stops (possibly due to an interrupt) or interconnect timing happens to allow enough time for one of them to complete.

NOTE: this erratum is describing additional limitations on exclusives loads and stores in a multi-core system including A15. There is no plan to fix this erratum on future A15 cores, as reasonable code following the ARM architecture guidelines should not be affected.

NOTE: This erratum matches bug #4637 in the ARM internal Jira database.

Conditions

1) One master continuously reading the location of the semaphore.
2) Two masters doing a ldrex/strex loop to the semaphore.
3) Semaphore in write-back shared memory.
4) Three master system (3+ core A15, or 3+ total processors in the system over ACE).

Implications

Neither C1 nor C2 will ever succeed in gaining the lock. Software could stop making progress. An interrupt to one of the cores C1/C2/C3 would likely break the livelock.

Workaround

If there are no more than two coherent masters in the system, no workaround is needed, the issue will not be seen.

The latest version of the ACE specification adds additional command types and system logic to allow processors to avoid this issue. This specification update was not available in time for A15 to take advantage of it and A15 does not implement this ACE feature. As an alternative, A15 installed hardware in each processor to detect that the load/store exclusive livelock scenario may be occurring and delay snoops for a period of time to allow the load exclusive/store exclusive loop to complete and make forward progress. With this fix, no existing code that uses ldrex/strex should need to be rewritten if it follows the ARM Architecture Reference Manual guidelines in the “A3.4 Synchronization and Semaphores” section and is not unreasonably long.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

To enable this hardware on Cortex-A15 you must set the "Snoop-delayed exclusive handling" bit in the Auxiliary Control Register, ACTLR[31] to 1. The reset value of ACTLR[31] is 0 for all product revisions r2pX, r3p0, r3p1 and r3p2.

Note: all references to “ldrex” encompass all Load-Exclusive instructions and “strex” encompass all Store-Exclusive instructions.

804969: Extension register loads and stores can livelock core

Category B Rare

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

In this erratum description, extension register means a register in the Advanced SIMD and floating-point register bank.

An Advanced SIMD or floating-point data processing instruction (OP1) can stall due to a constant stream of later Advanced SIMD or floating-point data processing instructions being speculatively issued and cancelled. These later operations are issued and cancelled because speculative extension register loads that provide operands for the later instructions are issued and cancelled. The loads are cancelled because they hazard against an extension register store to Strongly Ordered or Device memory. The extension register store cannot complete because the store data is being supplied by the stalled operation (OP1).

Under very specific timing conditions, it is possible for OP1 to be stalled indefinitely.

Note: This erratum matches bug #5451 in the ARM internal Jira database.

Conditions

1) Advanced SIMD or VFP data processing operation (OP1).

2) An extension register store (ST1) to Strongly Ordered or Device memory, dependent on OP1 results.

3) Three or more extension register loads after ST1 (LD1, LD2, LD3).

4) Two data processing operations dependent on each of LD1, LD2, LD3 (six operations in total).

5) Very specific timing.

Implications

In general, if all of the conditions are hit OP1 cannot to make forward progress. An interrupt will not break the stall. Any inner shared TLB invalidate instruction (e.g. TLBIMVAIS) to any address issued by another ARM core in the system will break the stall.

Extension register stores to Strongly Ordered or Device memory are not expected to be common in user applications.

This condition is currently only seen in contrived (non-useful) validation code. The erratum has not been seen in real applications.

Workaround

Avoid extension register stores to Strongly Ordered or Device memory.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

812169: DVM message blocked by copyback on AMBA Chi interconnect

Category B Rare

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

This erratum does not apply to any A15 that is using the standard ACE/AXI interconnect. It only applies to an A15 connected to the CCN-504 Chi interconnect through the SBAS bridge.

If a DVM transaction from the CCN-504 interconnect is sent down the L2 pipeline and the L2 Fill Evict Queue (FEQ) is full, the DVM transaction will be cancelled and reissued. If that DVM transaction also happens to have an address field that matches a copyback (WriteBack, WriteClean, or Evict) in the FEQ, the snoop logic might also detect a hazard that forces a dependency between the DVM and the copyback. Instead of immediately reissuing, the DVM transaction will be held off until the copyback completes.

On a general ACE interconnect this is not a problem, because copyback writes must make forward progress. However, in the Chi interconnect it is possible for that copyback to have a dependency on the DVM transaction completing. The copyback might not complete until the A15 completes the DVM request. This can lead to a system deadlock.

Note: This erratum matches bug #5458 in the ARM internal Jira database.

Configurations affected

Only affects A15 when used with CCN-504

Conditions

1) A DVM request (DVM1) is received on the AR channel from the CCN-504 interconnect.
2) DVM1 is processed when the FEQ is full.
3) DVM1 address matches the address of a WriteBack, WriteClean, or Evict (WB1) in the FEQ WB1 has started to issue, but has not completed sending out its data on the W channel.
4) WB1 is unable to issue its data due to back pressure from the bridge on the write channel. The back pressure will not release until DVM1 completes.

Implications

This erratum might very rarely lead to a system deadlock.

Workaround

To avoid this erratum, A15 configurations used with CCN-504 interconnect must set L2ACTLR[7]. This will enable a timeout mechanism which will allow the DVM request to be reissued even if the copyback has not completed.

When an A15 is used with the CCN-504 interconnect the L2ACTLR must be already configured as specified in the CCN-504 Technical Reference Manual. This workaround sets one additional bit in that register.

827923: TLB maintenance operations might not be synchronized by DSB instruction

Category B Rare

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

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-A15 the DSB instruction might complete before some store memory transactions with 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 #5493 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) Do not use write-back-no-allocate page table mappings.
2) Set ACTLR[26:25] = 2’b11 // disable write streaming no L1-allocate threshold.
3) Set ACTLR[28:27] = 2’b11 // disable write streaming no-allocate threshold.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

2.7. Category C

773023: Order may not be maintained between Strongly Ordered memory requests and Device memory requests on ACE/AXI

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

A15 maintains order between Strongly Ordered (SO) memory requests as required by the ARM architecture memory ordering model. This is done inside the A15 by use of internal ordering logic. On the interconnect, A15 enforces ordering by using the same ARID/AWID value for all SO memory requests from a given processor (guaranteeing read/read and write/write ordering) and by waiting for the completion of all SO and Device memory requests on one channel before issuing SO or Device requests from the same core on the other channel (guaranteeing read/write and write/read ordering).

A15 does the same for Device memory.

However, A15 uses different ARID and AWID for Device memory requests from a given CPU and Strongly Ordered memory requests from the same CPU. Due to this fact, it is possible that the an SO read from a given CPU could pass a Device read from the same CPU and arrive at a single peripheral out of order.

Conditions

1) A system has memory mapped peripherals larger than 4KB
2) Some of the pages mapped to that peripheral are mapped Strongly Ordered and some are mapped Device
3) Software depends upon ordering of these Strongly Ordered and Device memory requests

Implications

This is not an issue for any device that fits in one 4KB memory page, as it is only possible to have a single memory type for that page (SO/Dev aliasing is not allowed).

For larger peripherals, it is possible that Strongly Ordered or Device transactions could arrive at the peripheral out of order.

Workaround

A given peripheral device should be mapped to all Strongly Ordered or all Device memory.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

777769: ICache parity error may not be corrected for NC code

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

If an instruction fetch to a non-cacheable page in memory gets a false hit on a line in the instruction cache due to a parity error in the instruction cache tag array, incorrect instructions may be executed.

Note: This erratum matches bug #5312 in the ARM internal Jira database.

Conditions

1) MMU enabled
2) Instruction fetch to page marked SO/Dev/Normal-Non-Cacheable
3) Parity error in instruction cache tag
4) Corrupted tag matches the physical address of the cache line being fetched

Implications

In an A15 configured with L1 parity/ECC and with parity/ECC checking enabled, in very rare circumstances a parity error can cause delivery of bad instructions while executing non-cacheable code. This is not expected to be an issue in normal systems as no normal programs will have instructions in non-cacheable memory with the MMU enabled. At boot, or any other time that the MMU is disabled, the erratum will not occur.

Workaround

Place instructions in cacheable memory whenever possible. If you must run non-cacheable code with the MMU enabled, first invalidate the instruction cache.

784419: An unaligned load crossing a 4K boundary between SO/Dev memory and WB memory can stall the core until the next interrupt

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

If an unaligned load crosses a 4k page boundary between two pages where the lower page is mapped to Strongly-ordered or Device memory and the upper page is mapped to Normal, Write-Back Cacheable memory, in certain rare cases the core may stall until the next interrupt. A memory transaction that crosses a boundary between these page types is architecturally UNPREDICTABLE and should not occur in any real code. However, if this were to occur at the highest privilege level with interrupts disabled, it could deadlock that CPU.

Note: This erratum matches bug #5355 in the ARM internal Jira database.

Conditions

1) A load instruction is executed that accesses bytes from two different 4k pages
2) The lower page is Device or Strongly-ordered memory type
3) The upper page is Normal Write-Back Cacheable memory, that is also Read-Allocate, Write-Allocate, or both
4) The load instruction is one of the following:
    - LDRH or LDR
    - an LDM that is not 8-byte aligned
    - a VLD* to 32bit Sd registers that is not 8-byte aligned
    - a VLD* to 64bit Dd registers that is not 16-byte aligned

Implications

If the above conditions occur, it is possible that in rare cases the core will stall until the next interrupt. If this occurs in a situation where no interrupt will occur, the CPU deadlocks. Examples of where a deadlock might occur are if there is no timer interrupt in the system, or the conditions occur at the highest privilege level with interrupts disabled.

Workaround

Do not execute loads that cross between Strongly-ordered or Device memory and Normal memory pages. Architecturally, such a load is UNPREDICTABLE.

784469: CTI Authentication Status register is incorrect

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

The AUTHSTATUS register is a read-only register in the CTI that reports the debug level supported by the CTI and the current status of the debug level.

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

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

For each of these fields, the value of the status bits as returned by the CTI and their meanings are specified as below:

Value	Description
2'b10	Functionality disabled
2'b11	Functionality enabled

In the CTI each pair of bits ([3:2] and [1:0]) in the AUTHSTATUS register currently read:

- When functionality is disabled - 2'b01

but should read (as per table above):

- When functionality is disabled - 2'b10

The bits are swapped.

Condition 1

AUTHSTATUS[1:0] - Non-secure Invasive Debug

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

Condition 2

AUTHSTATUS[3:2] - Non-secure non-Invasive Debug

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

Implications

The status of the debug level supported by the CTI as returned by the AUTHSTATUS register read is incorrect. The masking of trigger inputs and outputs using DBGEN and NIDEN is not affected by this erratum. 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 read data accordingly.

788419: If a single memory address is aliased as both shareable and non-shareable, a CMO to this same line might deadlock

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2. Fixed in r3p3.

Description

If an invalidating data cache maintenance by MVA operation (DCIMVAC or DCCIMVAC) issued by an A15 CPU hits a dirty line in the L2 cache and collides in a narrow window with a ACE snoop to the same line, the data provided as part of the snoop response might be corrupted or the system could deadlock.

The cache maintenance operation allocates a data buffer to hold the dirty data to be evicted. The snoop misses in the cache, but is detected as a hit on the eviction that is in progress. The eviction data is converted to a snoop response, and the snoop becomes associated with the dirty data in the data buffer, instead of being associated with the cache maintenance operation data buffer entry. For one cycle during this conversion when the cache maintenance operation is assigning the data buffer to the snoop, that data buffer appears to be available for an unrelated new request. If a new operation that is in the pipeline during that one cycle is allocated to the data buffer, and this allocation overwrites the data before the data is read out for the snoop response, then the snoop data is corrupted.

Note: This erratum matches bug #5378 in the ARM internal Jira database.

Conditions

This erratum requires the following sequence of conditions:

1) The same physical memory address must be aliased as both shareable and nonshareable in two different page table entries.
2) A DCIMVAC or DCCIMVAC instruction is executed on an A15 CPU to cache line A.
3) Cache line A is in the L2 cache and is dirty.
4) An ACE snoop to line A occurs.
5) The ACE snoop misses the cache, because the invalidation has already occurred, but matches on the eviction in flight before the eviction is committed to the ACE write channel.
6) The ACE snoop data response is held up, but the cache maintenance operation for cache line A is issued on the AR channel, completes, and gets an R channel response.
7) The cache maintenance operation deallocates from the L2 cache buffers before the snoop data is sent out.

Implications

The erratum conditions require the CMO to be issued after the snoop has been received, but to complete before the snoop completes. This can occur when there is disagreement on the shareability of physical address A. A15 issues a CMO to address A as non-shared memory, but another master has issued a shared memory request to A which trigger the snoop. These transactions may go through different paths in the memory system and not hazard. Disagreeing on the shareability of memory is unpredictable in the ARM architecture and therefore the erratum cannot occur with correctly programmed page tables.

Workaround

For all systems:

- Ensure that all masters in the system agree on the shareability of all memory locations and that the interconnect will not issue snoops to A15 for non-shareable memory locations.

803620: Near zero Advanced SIMD fused multiply add may be incorrectly flushed to zero.

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

In very rare cases, a very small single-precision Advanced SIMD fused multiply add (VFMA/VFMS) that should have the result 2^-125 (two times the smallest nonzero number supported in Advanced SIMD) will instead be flushed to zero. Only fused multiply adds that should round to this very small value are affected, and not even all of those are affected.

Note: This erratum matches bug #5438 in the ARM internal Jira database.

Conditions

1) Advanced SIMD FMA instruction is executed.
2) Correct result of the fused multiply add is 2^-125.
3) The product and the augend have opposite signs.
4) The fraction for the sum is exactly zero with a carry out.

Implications

In the rare case that the erratum conditions are hit, the result will be zero rather than 2^-125. Advanced SIMD already rounds any number less than 2^-126 to zero. It is not expected that this erratum will have a noticeable effect in typical applications.

Because Advanced SIMD already rounds subnormals to zero, it is expected that most code that cannot tolerate the rounding of extremely small numbers to zero will be using the VFP versions of these instructions (which are not affected by the erratum).

Workaround

This erratum only applies to the Advanced SIMD version of the VFMA/VFMS. The VFP versions of VFMA and VFMS return the correct result. Code that cannot tolerate the rounding of very small results to zero should use the VFP versions of VFMA and VFMS

803670: Unaligned load accessing two cache lines could return uncorrected data in case of single bit ECC error in the L2 cache

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

If a load micro-operation accesses memory in two cache lines, misses the L1 cache for both cache lines, and hits a single-bit ECC error in the L2 for the first cache line, uncorrected data can be returned.

If ECC is not configured or is not enabled, the erratum is not possible.

Note: This erratum matches bug #5423 in the ARM internal Jira database.

Conditions

1) An unaligned load is executed that touches two 64-byte aligned regions and is one of the following:
   - LDRH or LDR
   - An LDM that is not 8-byte aligned
   - VLD* to 32bit Si registers that is not 8-byte aligned
   - VLD* to 64bit Di registers that is not 16-byte aligned
2) Both 64-byte cache lines miss in the L1 cache.
3) The lower cache line hits the L2 cache with a single bit ECC error in the data array.
4) At the point the bad data returns all previous loads in the machine have completed.

Implications

If the erratum conditions occur, uncorrected data can be returned for the load and no abort is signalled. The L2 memory error syndrome register is correctly updated to reflect a single bit ECC error. The data will be corrected in the L2 cache and future loads will return correct data.

Although unaligned loads that cross 64-byte aligned boundaries can occur, they are expected to be much less common than aligned transactions.

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.

803671: The ACE ARDOMAIN field may be incorrect on some Instruction or Tablewalk prefetch requests

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

The A15 will prefetch cache lines that are expected to be needed in the near future based on current instruction fetches and translation table walks. In certain rare cases, a prefetch request for a table walk cache line might use the shareability attributes from a recent instruction fetch request, or an instruction prefetch might use the shareability attributes from a recent table walk. If this occurs and the prefetch request misses in the L2 cache, a cache line read request is issued on the ACE AR channel with an incorrect ARDOMAIN field. This might lead to incorrect snooping behavior in the ACE interconnect.

If a prefetch read is issued with higher shareability than it should have (outer-shareable rather than inner-shareable or non-shareable, inner-shareable rather than non-shareable), there will be no functional impact, although very rarely an unnecessary snoop might be issued. However, if a prefetch read is issued with lower shareability than it should have, there could be a coherence issue. The prefetch request might be issued as ReadNoSnoop and not correctly snoop other caches in the system.

Note: This erratum matches bug #5441 in the ARM internal Jira database.

Configurations Affected

ACE Systems only

Conditions

1) Instruction prefetch or table walk descriptor prefetch are enabled, because either:
    - L2PFR[8:7] is not zero
    - OR L2PFR[10] is zero
2) Instruction fetch requests (IF request) and translation table walk descriptor fetches (TBW request) are both being issued,
3) The IF requests and TBW requests have different shareability attributes.
4) Rare timing boundary conditions occur.

Implications

In general, this is not expected to be a problem in most systems. In a non-ACE system, the errata cannot occur (because A15 snoops all L1 caches for all cacheable requests regardless of shareability). In an ACE system, it is expected that code and translation tables will exist in cacheable memory regions with consistent shareable attributes (either outer-shareable or inner-shareable).

Workaround

To avoid this erratum, use the same shareability attribute (non-shareable, inner-shareable, or outer-shareable) for all translation table memory and executable instruction memory. This is the preferred workaround.

Alternatively, the erratum can be avoided by disabling both instruction prefetch and table walk descriptor prefetch (by setting L2PFR[10] = 1 and L2PFR[8:7] = 0). This may cause some performance degradation depending on the application.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

808469: Deprecated SWP/SWPB can cause a stall until the next interrupt

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2 Fixed in r3p3.

Description

In the r2pX and r3p2 revisions of A15 with the ECO for 798181 applied (REVIDR[4]=1), if the SCTLR.SW bit (SWP and SWPB enable) is set to 1 it is possible in a very rare case for a SWP or SWPB instruction to stall until the next interrupt. SWP and SWPB are deprecated instructions and ARM strongly recommends not using them. Generally, the OS leaves SCTLR.SW at its reset value of 0 and this erratum will not apply.

Note: This erratum matches bug #5457 in the ARM internal Jira database.

Configurations affected

Must have Errata 798181 ECO, REVIDR[4] set.

Conditions

1) SCTLR.SW set to 1 (enabling SWP/SWPB).
2) SWP/SWPB instruction executed.
3) The load portion of the swap instruction executes and returns data ahead of an older load.
4) The store portion of the swap instruction is unable to proceed due to write congestion.
5) A DVM synch message is received.

Implications

If the above conditions occur, the core will be unable to respond to the DVM synch request or to complete the SWP/SWPB instruction. The core will stall indefinitely. The next interrupt or debug exception will flush the SWP instruction and execution will proceed normally.

In a typical system, the SCTLR.SW bit will be 0 and this erratum will not apply.

Workaround

To avoid this erratum, leave SCTLR.SW=0 on affected versions of A15, and either avoid SWP/SWPB or allow the OS to emulate the instructions.

810072: When a single-bit ECC error occurs in the L2, uncorrected data might be returned

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

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.

This erratum cannot occur if ECC is not configured or is not enabled.

Note: This erratum matches bug #5464 in the ARM internal Jira database.

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. Write streaming is enabled (ACTLR[28:25] != b1111) in the L1.
3) A full cache line-streaming write to address B (WR_B) is ready to issue to the L2.
4) 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.

816469: Double-bit ECC error during hardware correction of a single-bit ECC error might cause data corruption

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

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 #5471 in the ARM internal Jira database.

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.

822719: Read following a write of a Timer TVAL register might return incorrect value

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

In Cortex-A15, 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 #5480 in the ARM internal JIRA database.

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.

826375: Debug accesses in User mode do not properly generate undefined instruction exceptions for some SIMD and VFP registers

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

Accessing any of the Advanced SIMD and Floating-point Extension system registers (VMRS FPSID, VMRS MVFR0, VMRS MVFR1, VMRS/VMSR FPSCR, VMRS/VMSR FPEXC) while in User mode and in Debug state should generate an Undefined Instruction exception. Instead, these instructions are executed.

Note: This erratum matches bug #5488 in the ARM internal JIRA database.

Conditions

1) The processor is in User mode and Debug state.
2) Execution of any of these instructions issued through the Instruction Transfer Register (DBGITR):
    - VMRS FPSID
    - VMRS MVFR0
    - VMRS MVFR1
    - VMRS FPEXC
    - VMSR FPEXC
    - VMRS FPSCR (with FPEXC.EN==0)
    - VMSR FPSCR (with FPEXC.EN==0)

Implications

If the above conditions occur, the processor will perform the specified system register accesses instead of generating an Undefined Instruction exception.

(When FPSCR is accessed from User mode with FPEXC.EN==1 the expected behavior is to perform the access, and Cortex-A15 MP Core does behave that way).

Workaround

There is no workaround.

826969: Cortex-A15 might violate read-after-read memory ordering on a load forwarding from a store crossing a 16-byte boundary

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3.

Description

Cortex-A15 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 #5491 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 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 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 characterized as being Category C. Additionally, this erratum has been present in volume shipping devices (Cortex-A15) without being observed in the field.

Workaround

No workaround is necessary.

Copyright © 2015 ARM. All rights reserved.

Non Confidential

832972: HSTR.{T7,T8,T15} bits incorrectly trap CDP instructions

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3. Open.

Description

HSTR.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 #5496 in the ARM internal JIRA database.

Configurations affected

All configurations are affected.

Conditions

1) HSTR.Tx is set to 1.
2) A Non-secure CDP or CDP2 instruction with CRn set to x is executed in PL1 or PL0 where x = {7,8,15}.

Implications

If the above conditions are met, the CDP/CDP2 instruction will be erroneously trapped to Hyp mode. The correct behavior is to take an UNDEF exception.

Workaround

There is no workaround.

834569: PMU event BUS_CYCLES might be incorrect in some cases

Category C
Products Affected: Cortex-A15 MP Core -NEON.
Present in: r3p0, r3p1, r3p2, r3p3. Open. .

Description

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

Note: This erratum matches bug #5497 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.

842119: Instruction issued through ITR by debugger executes incorrectly for PCLKDBG frequencies much slower than the processor

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3. Open.

Description

When the processor is in Debug state, the debugger can execute instructions on the processor using Instruction Transfer Register, DBGITR. Under certain conditions when the PCLKDBG clock frequency is considerably slower than the processor clock frequency, the processor might incorrectly execute the same instruction more than once.

Note: This erratum matches bug #5501 in the ARM internal Jira database.

Configurations affected

All configurations where PCLKDBG clock frequency is slower than one fourth of processor clock frequency are affected.

Conditions

1) The processor is currently in Debug state.
2) Debug Status and Control Register, DBGDSCR.ExtDCCmode[1:0] is set to 0b01, that is, Stall mode.
3) The external debugger transfers an ARM instruction to the processor for execution by writing to Instruction Transfer Register, DBGITR.

Implications

If the above conditions are met, then the processor might incorrectly execute the contents of the Instruction Transfer Register, DBGITR more than once. This can have unintended side effects, for example additional memory transactions from the processor or incorrect execution of store exclusive instructions.

Workaround

Software Workaround: The debugger needs to avoid using Stall mode (DBGDSCR.ExtDCCmode[1:0]=0b01) and alternately use Non-blocking mode (DBGDSC.ExtDCCmode[1:0]=0b00) or Fast mode (DBGDSCR.ExtDCCmode[1:0]=0b10) when an ARM instruction is transferred to DBGITR for execution on the processor.

Hardware Workaround: Run the PCLKDBG clock frequency faster than or equal to one fourth of the processor clock frequency.

851024: Persistent evictions combined with interconnect backpressure might stall Write-Back No-Allocate stores

Category C

Products Affected: Cortex-A15 MP Core -NEON.

Present in: r3p0, r3p1, r3p2, r3p3. Open.

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 #5504 in the ARM internal tracking system

Configurations affected

Coherent ACE systems that enable WriteUnique/WriteLineUnique transactions.

Conditions

1) WriteUnique/WriteLineUnique transactions are enabled by setting L2ACTLR[4] to 1’b0. This is the reset value.
2) A Cortex-A15 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.

Workaround

If WriteUnique/WriteLineUnique transactions are not required, disable them by setting L2ACTLR[4] = 1'b1. 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.

Copyright © 2015 ARM. All rights reserved.

Non Confidential