Arm® Cortex®-R7 MPCore

Product Revision r0

Software Developers Errata Notice

Non-Confidential - Released

Software Developers Errata Notice

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ

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, PJDOC-466751330-10082
- 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.

15 Mar 2019: Changes in Document v5

Page	Status	ID	Cat	Rare	Summary of Erratum
41	New	1414441	CatC		The debugger might hang in Lock mode if it accesses CPU1 registers

23 Mar 2016: Changes in Document v4

Page	Status	ID	Cat	Rare	Summary of Erratum
29	New	854324	CatB		Write access to the SCU Control Register may result in unexpected value
30	New	854436	CatB		CP15 write access to Cache Tag RAM location might lead to data corruption
31	New	854437	CatB		In Low Latency Interrupt mode, an IRQ or a FIQ received might generate a data abort
32	New	854722	CatB		Broadcasting invalidate instruction Cache by MVA may stall a core
38	New	854271	CatC		CP15 read access to ITCM RAM location might return incorrect data value
39	New	854272	CatC		UNDEFINED instructions might not raise an UNDEF exception
40	New	854771	CatC		Eviction request missing in tag RAM can incorrectly detect an ECC error in Data RAM

07 Feb 2013: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
37	New	800669	CatC		DBGPRSR Sticky Reset status bit is set to 1 by the CPU debug reset instead of by the CPU non-debug reset

27 Sep 2012: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
11	New	786372	CatA		Conditional Load in Normal Memory Non-cacheable region might deadlock the processor
12	New	794771	CatA		FPU load or store data might be corrupted
13	New	794772	CatA		Running the processor with ECC enabled might cause data corruption
13	New	783369	CatB		In Low Latency Interrupt mode, an IRQ received while handling an FIQ might deadlock the processor
15	New	783519	CatB		Diagnostic information in ECC Error bank might be wrong
16	New	783520	CatB		Stores in Normal, Non-cacheable, Shareable memory-regions might generate two external writes and so cause a synchronization issue
18	New	787180	CatB		On executing an LDMIA instruction to the PC, an extra Conditional Flush element is generated, producing incorrect conditional trace
19	New	787921	CatB		In Low Latency Interrupt mode, an IRQ or an FIQ received while executing a divide instruction might deadlock the process
20	New	788769	CatB		Store execution time to Normal Non-cacheable, Strongly-ordered or Device memory regions is not bounded and might livelock the processor
22	New	790769	CatB		Incorrect data value trace on VLDR or VLDM instructions
23	New	794670	CatB		In Split-Lock “safe” mode, the SCU Configuration register does not report the correct number of processors
24	New	794672	CatB		DSB/DMB in Low Latency Interrupt mode might deadlock the processor
25	New	794673	CatB		Dynamically disabling the FPU might deadlock the processor
26	New	794729	CatB		The QoS feature might cause data corruption when data coherency is enabled

27	New	794773	CatB	A short loop with DMB might prevent a CP15 broadcast operation making forward progress, causing a denial of service
28	New	794971	CatB	Data values might not be traced
33	New	786371	CatC	Accesses to a Normal Memory Cacheable region going to the AXI peripheral port might deadlock the processor
34	New	787175	CatC	Data values might be missing from data trace when dynamic clock gating is enabled
35	New	793269	CatC	“Write Context ID” event is updated on read access
36	New	794719	CatC	Performance monitor counters continue counting after being disabled

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-R7 MPCore processor.

1.1. Scope of this document

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

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

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

1.2. Categorization of errata

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

Table 1 Categorization of errata

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

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

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

rn Identifies the major revision of the product.

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

2.2. Revisions Affected

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

This document includes errata that affect revision r0 only.

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

Table 2 Revisions Affected

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1
794772	CatA		Running the processor with ECC enabled might cause data corruption	X	
794771	CatA		FPU load or store data might be corrupted	X	
786372	CatA		Conditional Load in Normal Memory Non-cacheable region might deadlock the processor	X	
854722	CatB		Broadcasting invalidate instruction Cache by MVA may stall a core	X	X
854437	CatB		In Low Latency Interrupt mode, an IRQ or a FIQ received might generate a data abort	X	X
854436	CatB		CP15 write access to Cache Tag RAM location might lead to data corruption	X	X
854324	CatB		Write access to the SCU Control Register may result in unexpected value	X	X
794971	CatB		Data values might not be traced	X	
794773	CatB		A short loop with DMB might prevent a CP15 broadcast operation making forward progress, causing a denial of service	X	
794729	CatB		The QoS feature might cause data corruption when data coherency is enabled	X	
794673	CatB		Dynamically disabling the FPU might deadlock the processor	X	
794672	CatB		DSB/DMB in Low Latency Interrupt mode might deadlock the processor	X	
794670	CatB		In Split-Lock “safe” mode, the SCU Configuration register does not report the correct number of processors	X	
790769	CatB		Incorrect data value trace on VLDR or VLDM instructions	X	

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1
788769	CatB		Store execution time to Normal Non-cacheable, Strongly-ordered, or Device memory regions is not bounded and might livelock the processor	X	
787921	CatB		In Low Latency Interrupt mode, an IRQ or an FIQ received while executing a divide instruction might deadlock the process	X	
787180	CatB		On executing an LDMIA instruction to the PC, an extra Conditional Flush element is generated, producing incorrect conditional trace	X	
783520	CatB		Stores in Normal, Non-cacheable, Shareable memory-regions might generate two external writes and so cause a synchronisation issue	X	
783519	CatB		Diagnostic information in ECC Error bank might be wrong	X	
783369	CatB		In Low Latency Interrupt mode, an IRQ received while handling an FIQ might deadlock the processor	X	
1414441	CatC		The debugger might hang in Lock mode if it accesses CPU1 registers	X	X
854771	CatC		Eviction request missing in tag RAM can incorrectly detect an ECC error in Data RAM	X	X
854272	CatC		UNDEFINED instructions might not raise an UNDEF exception	X	X
854271	CatC		CP15 read access to ITCM RAM location might return incorrect data value	X	X
800669	CatC		DBGPRSR Sticky Reset status bit is set to 1 by the CPU debug reset instead of by the CPU non-debug reset	X	X
794719	CatC		Performance monitor counters continue counting after being disabled	X	
793269	CatC		“Write Context ID” event is updated on read access	X	
787175	CatC		Data values might be missing from data trace when dynamic clock gating is enabled	X	
786371	CatC		Accesses to a Normal Memory Cacheable region going to the AXI peripheral port might deadlock the processor	X	

2.3. Category A

786372: Conditional Load in Normal Memory Non-cacheable region might deadlock the processor

Category A

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The Cortex-R7 Load-Store unit has four slots. A load instruction might use several of these slots in the Load-Store Unit. This erratum arises when a conditional load to a Non-cacheable memory region that uses at least two slots, and the first slot receives an abort, whilst the others do not, fails its condition codes. This has to be followed by another load, to a Normal Non-cacheable memory region which uses an available slot in the Load-Store Unit. If the data requested by the second load is returned before the processor has resolved the abort condition of the first load, then a data corruption might occur within the Load-Store Unit slots and this might lead to a processor deadlock.

Configuration affected

The erratum affects all configurations.

Conditions

The erratum requires the following conditions:

- A conditional load is executed in a Normal, Non-cacheable memory region, where:
  - The condition of this load is failing.
  - This load uses two or three Load-Store Unit Slots. For example, when <c> is not AL, an LDR<c> targeting byte 5, 6 or 7; or an LDRH<c> targeting byte 7; or an LDM2<c>
  - The first slot aborts, the second slot does not. The concerned aborts can be:
    - An alignment fault (For example, an LDR<c> targeting byte 5 when SCTLR.A bit is set to 1; or an LDM2<c> that is not word-aligned)
    - A watchpoint debug event (For example, the first slot hits a watchpoint, the second slot doesn’t)
    - A permission fault (For example, an LDM2<c> that crosses two MPU regions, the first slot hits a MPU region with no access rights, the second slot hits a MPU region with read access rights)
    - A background fault memory (if background region is disabled, the first slot doesn’t hit any MPU region and the second one hits a MPU region with read access rights).
- Another load is executed in a Normal Non-cacheable memory region
- Under certain timing circumstances, the data requested by the second load is returned before the processor has resolved the abort condition of the first load.

Note that this erratum can also apply to Normal, Write-Through Cacheable memory regions, because Write-Through attributes is remapped to Non-cacheable.

Implications

The erratum might cause data corruption in the Load-Store Unit slots and therefore might lead to a processor deadlock.

Workaround

There is no work-around.

794771: FPU load or store data might be corrupted

Category A

Products Affected: Cortex-R7 FPU.

Present in: r0p0

Description

Under rare timing circumstances, floating-point load or store can be corrupted.

Configurations affected

The erratum affects all configurations that use the optimized FPU.

Conditions

The erratum might happen under the following conditions:

- A floating-point load or store instruction is executed, and is flushed because it is placed after a mis-predicted branch or after an abort exception.
- A floating-point load or store instruction is executed.

Implications

The erratum might cause a data corruption.

Workaround

There is no workaround for this erratum.

794772: Running the processor with ECC enabled might cause data corruption

Category A

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

Under certain timing circumstances when ECC is enabled on the data cache or on the data TCM, data corruption might happen.

Configurations affected

The erratum affects all configurations that include the ECC.

Conditions

The erratum can happen when ECC is enabled on the data cache or on the data TCM, that is when ACTLR bit [9] is set to 1.

Implications

The erratum might cause a data corruption.

Workaround

There is no workaround for this erratum. ECC must remain disabled.

2.4. Category A (Rare)

There are no errata in this category

2.5. Category B

783369: In Low Latency Interrupt mode, an IRQ received while handling an FIQ might deadlock the processor

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

A processor deadlock might happen in Low Latency Interrupt mode, when some CP15 cache maintenance operations are used, and when an IRQ is received while handling an FIQ.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum can happen under the following conditions:

- The Low Latency Interrupt mode is enabled, by setting SCTLR.FI to 1
- The processor executes a CP15 cache operation maintenance (that is, any p15, 0, <Rd>, c7, <cRm>,<opc2> operation except p15,0, <Rd>, c7, c5,4 (ISB), p15,0,<Rd>, c7, c10,4 (DSB), or p15,0, <Rd>, c7, c10,5 (DMB))

- An FIQ occurs.
- Because the low latency mode is enabled, the hardware discards the CP15 cache maintenance operation automatically, and then the FIQ is taken by the processor.
- Before the CP15 cache operation maintenance is discarded, an IRQ also occurs.

As a consequence, any subsequent CP15 operations are automatically discarded by the hardware, and this might lead to a processor deadlock. Also any subsequent IRQ cannot be taken, unless the processor enters debug state.

Implications

The erratum might cause a processor deadlock, and prevent subsequent entry into any IRQ.

Workaround

The recommended workaround requires setting bit [24] (Disable discarding of cp15 cache maintenance operations in Low Latency Interrupt mode) in the undocumented CP15 Diagnosis register by the following read-modify-write sequence:

```
MRC     p15,0,r0,c15,c0,1   ; read undocumented CP15 Diagnosis register
ORR     r0,r0,#(1<<24)
        MCR     p15,0,r0,c15,c0,1   ; write undocumented CP15 Diagnosis register
```

783519: Diagnostic information in ECC Error bank might be wrong

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

When an ECC error occurs on the Data Cache Data RAM, the faulty word affected might be wrongly reported in the ECC Error Bank Registers (DEER0-2). The index of the RAM bank is provided, rather than the index of the word. This means the diagnostic information provided to a safety system might be erroneous.

Configurations affected

The erratum affects all configurations that use ECC.

Conditions

The erratum requires the following conditions:

- The ECC is enabled on the caches (because bit [9] of Auxiliary Control Register, ACTLR is set to 1)
- An ECC error occurs on the Data Cache Data RAM

Implications

The erratum might cause the safety system to get erroneous diagnostic information.

Workaround

The recommended workaround for this erratum is to translate the bank information into word information. The correct word index can be calculated from the bank index and way number using:

```
word_index = (bank_index - 2 * way) & 0x7
```

For example, the following assembly code can be used to correct the value of a DEERn register held in r0:

```
CLZ   r1,r0                 ; Find flagged way from DEER[31:28]
      RSB   r1,r1,#3        ; Convert to a way number
UBFX  r2,r0,#2,#3           ; Extract DEER[4:2] bank index
      SUB   r1,r2,r1,LSL#1  ; Compute word index (pre modulo 8)
      BFI   r0,r1,#2,#3     ; Replace DEER[4:2] with word index
```


783520: Stores in Normal, Non-cacheable, Shareable memory-regions might generate two external writes and so cause a synchronisation issue

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The Cortex-R7 processor buffers stores in a store buffer before writing them to the AXI bus. The following description refers to a naturally aligned doubleword memory location.

A non-Exclusive store to Normal, Non-cacheable, Shareable memory is being written from the store buffer to the AXI bus, and a subsequent non-Exclusive store to one or more bytes within the same naturally aligned 64-bit memory location as the first store is written to the store buffer, and the subsequent store is not an exact repeat of the first store, and there is no DMB between these two stores. Due to the merging policy of the Cortex-R7 store buffer, the first write might be merged with the second one, and it might be sent twice to the same location on the AXI bus.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum requires the following conditions:

- A non-Exclusive store is executed in Normal, Non-cacheable, Shareable memory.
- Another non-Exclusive store is executed in one or more bytes of the same naturally aligned 64-bit memory location, but not in exactly the same bytes as the original store.

In this case, and under certain timing circumstances, the first store might be written twice to the external memory.

Note that this erratum can also apply to Normal, Write-Through Cacheable Shareable memory regions, because Write-Through attributes is remapped to Non-cacheable

Implications

This erratum might have implications in a multi-master system where control information is passed between several processing elements in memory using a communication variable, for example a semaphore. In such a system, it is common for communication variables to be claimed using a Load-Exclusive/Store-Exclusive, but for the communication variable to be cleared using a non-Exclusive store. This erratum means that the clearing of such a communication variable might occur twice. This might lead to two masters apparently claiming a communication variable, and therefore might cause data corruption to shared data.

A scenario in which this might happen is:

```
        MOV     r1,#0x40          ; address is double-word  aligned, mapped in
                                    ; Normal Non-cacheable Shareable memory
Loop:   LDREX   r5, [r1,#0x0]     ; read the communication variable
        CMP     r5, #0            ; check if 0
        STREXEQ r5, r0, [r1]      ; attempt to store new value
        CMPEQ r5, #0              ; test if store succeeded
        BNE Loop                  ; retry if not
        DMB                       ; ensures that all subsequent accesses are observed when
                                   ; gaining of the communication variable has been observed
                                  ; loads and stores in the critical region can now be performed
        MOV     r2,#0
        MOV     r0, #0
        DMB                       ; ensure all previous accesses are observed before the
```



```
                                ; communication variable is cleared
STR     r0, [r1]        ; clear the communication variable with normal store
STR     r2, [r1,#0x4]   ; previous STR may merge under certain timing circumstances
                        ; and thus may be sent again. This causes undesired release
                                ; of the communication variable.
```

This scenario is valid when the communication variable is a byte, a half-word, or a word

Workaround

There are several possible workarounds:

1. Add a DMB after clearing a communication variable:

```
STR     r0, [r1]        ; clear the communication variable
DMB                     ; ensure the previous STR is complete
```

Also any IRQ or FIQ handler must execute a DMB at the start to ensure as well the clear of any communication variable is complete.

2. Ensure there is no other data using the same naturally aligned 64-bit memory location as the communication variable:

```
                        ALIGN 64
communication_variable  DCD 0
unused_data                     DCD 0
                        LDR r1,= communication_variable
```

3. Use a Store-Exclusive to clear the communication variable, rather than a non-Exclusive store.

787180: On executing an LDMIA instruction to the PC, an extra Conditional Flush element is generated, producing incorrect conditional trace

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The Cortex-R7 ETM supports conditional instruction trace, which can be enabled in the trace configuration register. When conditional load trace is enabled, if the processor executes a conditional LDMIA instruction that includes the PC, under certain conditions, conditional trace might become incorrect. Conditional trace is decompressible prior to the LDMIA instruction, but after the LDMIA instruction the conditional trace stream is corrupted.

Instruction and data trace are not affected.

Configurations affected

The erratum affects all configurations that include and use the ETM.

Conditions

The erratum can happen under the following conditions:

- Conditional trace of load instructions is enabled by setting TRCCONFIGR.COND to a value of 0x1 or 0x3 or 0x7 (ETM-R7 Trace Configuration register).
- A conditional LDMIA instruction that includes the PC is executed
- In this case, under certain timing circumstances, the conditional trace might be not correct.

Implications

The trace stream includes an additional Conditional Flush element that makes the conditional instruction trace incorrect after the LDMIA instruction. Conditional instruction trace might remain incorrect for several instructions following the LDMIA instruction. However, conditional instruction trace recovers on the next Conditional Flush element, and is correct after this element.

Workaround

There is no reliable detection mechanism for this erratum. There is also no workaround for this erratum.

787921: In Low Latency Interrupt mode, an IRQ or an FIQ received while executing a divide instruction might deadlock the process

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

SDIV and UDIV instructions generate an Undefined Instruction exception on a divide-by-zero when SCTLR.DZ is set to 1.

When SCTLR.DZ is set, and when a divide instruction is executed with a zero divisor in Low Latency Interrupt mode, if an interrupt occurs the processor might deadlock.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum can happen under the following conditions:

- The Low Latency Interrupt mode is enabled, by setting SCTLR.FI to 1.
- Divide by zero fault detection is enabled, by setting SCTLR.DZ to 1.
- The processor executes a UDIV or SDIV instruction with a null divisor.
- An IRQ or an FIQ occurs.

Because the low latency mode is enabled, the hardware should discard the divide instruction, meaning it should not take the Undefined Instruction exception. However the processor still enters the Undefined Instruction exception and this might cause the processor to deadlock.

Implications

The erratum might cause a processor deadlock.

Workaround

The recommended workaround for this erratum is not use the Divide by zero fault detection, by setting SCTLR.DZ to 0. Another workaround is to test the divisor is not null before executing the divide instruction.

788769: Store execution time to Normal Non-cacheable, Strongly-ordered or Device memory regions is not bounded and might livelock the processor

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The Cortex-R7 Load-Store unit of each processor orders accesses to Normal Non-cacheable memory regions independently of the ordering of accesses to Strongly-ordered and Device memory regions. However the Cortex-R7 Data Bus Interface Unit of each processor does not make this distinction and orders all accesses regardless of whether they access Normal Non-cacheable memory regions or Strongly-ordered or Device memory regions. To cope with this difference between the Load-Store unit and the Data Bus Interface Unit:

- A read to Normal Non-cacheable memory always has priority over a store to Strongly-ordered or Device memory.
- A read to Strongly-ordered or Device memory always has priority over a store to Normal Non-cacheable memory.
- As a result, store execution time is not bounded. Under some circumstances this might live lock the processor.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum might happen under the following sequence of conditions:

- The processor executes a store to a Normal, Non-cacheable memory region.
- The processor then executes a succession of loads from Strongly-ordered or Device memory regions.

In this case the store is not executed until all the loads have been issued, which has an impact on the execution time of the store.

The erratum can also happen if the store is to a Strongly-ordered or Device memory region, and the loads are from Normal, Non-cacheable memory regions.

Implications

This erratum might have implications in a multi-master system where control information is passed between several processing elements in memory using a communication variable, for example a semaphore. In such a system, it is common for communication variables to be claimed using a Load-Exclusive/Store-Exclusive, and for the communication variable to be cleared using a non-Exclusive store. The following example describes the steps of one scenario:

1) CPU0 is claiming a communication variable (COM_VAR) in Normal Non-cacheable memory.
2) CPU0 is clearing the COM_VAR communication variable in Normal Non-cacheable memory, using a store.
3) CPU0 is waiting for the result of a variable (PERIPH_VAR) that is held in Device memory space and is accessed with a read within a loop.
4) CPU1, or any other master, is claiming the COM_VAR communication variable.
5) CPU1, or any other master, is clearing the COM_VAR communication variable.
6) CPU1, or any other master, is setting PERIPH_VAR in Device memory space.

This erratum causes step 2 to be blocked by step 3, additionally:

- Step 3 is blocked by step 6, that must come after steps 5 and 4
- Step 4 is blocked by step 2.

This means step 3 is blocked by step 2. So it causes the processor to livelock.

An interrupt will break the livelock.

Workaround

The recommended workaround for this erratum is to execute a DMB:

- Between a store in Normal Non-cacheable memory, and a load in Strongly-ordered or Device memory.
- Between a store in Strongly-ordered or Device memory, and a load in Normal Non-cacheable memory.

790769: Incorrect data value trace on VLDR or VLDM instructions

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

ETM data value trace is enabled using the TRCCONFIGR.DV register field. When data value trace is enabled, data addresses and data values are traced as they occur.

When data value trace is enabled, the data value of a VLDM or VLDR instruction might be traced incorrectly. The data address of the VLDM/VLDR instruction is traced correctly. All other data address and data value pairs are not affected. The data trace stream remains completely decompressible.

The instruction trace stream is not affected.

Configurations affected

The erratum affects all configurations that include and use the ETM.

Conditions

The erratum can occur if all of the following conditions are met:

- Data value tracing is enabled in the ETM, by setting TRCCONFIGR.DV bit to 1. A VLDM or VLDR instruction is executed by the processor.
- The data address of the VLDM or VLDR is word aligned but not double-word aligned.

Implications

The traced data value of the VLDM or VLDR instruction might be incorrectly traced. If data address[2:0] == 3’b100 (word aligned), traced value will correspond to data at address[2:0]==3’b000 (double-word aligned).

Workaround

There is no workaround for this erratum.

794670: In Split-Lock “safe” mode, the SCU Configuration register does not report the correct number of processors

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The SCU Configuration register is a read-only register that indicates the number of processors present in the Cortex-R7 MPCore processor. When the Cortex-R7 MPCore processor has been built as Split-lock and the “safe” mode is enabled, this register must indicate that only one processor is present. However, this erratum means this register indicates two processors are present.

Configurations affected

The erratum affects configurations that have two processors built as Split-lock, when safe mode is enabled.

Conditions

The erratum can happen under the following conditions:

- The Safe Mode has to be enabled, by asserting the SAFEMODE input signal HIGH.
- The processor reads the SCU Configuration register bit[0], to get the number of processors present.

In this situation, the value read indicates two processors are present, when it should indicate that only one is present.

Implications

Typically an operating system might get the wrong number of processors, and so this erratum might cause an OS boot failure.

Workaround

The recommended workaround requires accessing the Auxiliary Control Register. When the Cortex-R7 MPCore processor is built as Split-lock and the “safe” mode is enabled, the ACTLR.FW bit is RAZ/WI.

This means software can determine whether the processor is built as split-lock and has safe mode enabled as follows:

1. Write 0b1 to the ACTLR.FW bit.

2. Read the ACTLR register:

    - If ACTLR.FW is 0b0, “safe” mode is enabled.
    - If ACTLR.FW is 0b1, “performance” mode is enabled.

To do this, the following code sequence can be executed in privileged mode:

```
        MRC p15, 0, r0, c1, c0, 1       ; Read ACTLR into r0
        ORR r0, r0, #1                          ; Write 1 to bit 0 of r0
        MCR p15, 0, r0, c1, c0, 1       ; Write r0 to ACTLR, this tries
                                        ; to set ACTLR[0] to 1

        MRC p15, 0, r1, c1, c0, 1       ; Read again ACTLR into r1
        AND r1, r1, #1                          ; Mask all bits except bit 0
        CMP r1, #1                      ; Compare r1 with constant value 1
        BNE Safe
Perf:   <code>                          ; Performance-mode code
Safe:   <code>                          ; safe-mode code
```


794672: DSB/DMB in Low Latency Interrupt mode might deadlock the processor

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

In Low Latency Interrupt mode, DSB or DMB instructions executed after some floating-point stores are discarded when an IRQ or an FIQ is received. However this erratum means the data of the floating-point store might never be transferred to the Load-Store unit, which might cause the processor to deadlock.

Configurations affected

The erratum affects configurations that include the full FPU or the optimized FPU.

Conditions

The erratum can happen under the following conditions:

- The Low Latency Interrupt mode is enabled, by setting SCTLR.FI to 1
- The processor executes a floating point store, VSTR or VSTM
- The processor performs a DSB or a DMB operation
- An IRQ or an FIQ occurs.

Because the low latency mode is enabled, the hardware discards the DSB or DMB automatically.

Under certain timing circumstances, the data of the floating-point store might never be transferred to the Load-Store unit, and this might cause a processor deadlock.

Implications

The erratum might cause a processor deadlock.

Workaround

The recommended workaround requires setting bit [23] (Disable discarding of DSB/DMB in Low Latency Interrupt mode) in the undocumented CP15 Diagnosis register by the following read-modify-write sequence:

```
MRC    p15,0,r0,c15,c0,1    ; read undocumented CP15 Diagnosis register
ORR    r0,r0,#(1<<23)              ; set bit[23] of register r0 to 1
MCR    p15,0,r0,c15,c0,1    ; write undocumented CP15 Diagnosis register
```

794673: Dynamically disabling the FPU might deadlock the processor

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The Floating-Point Exception register (FPEXC) or the Coprocessor Access Control register (CPACR) might be used to control the clock-gating of the FPU. However, under unusual timing circumstances, disabling the FPU might discard some FPU instructions stalled in early stages of the core pipeline. This might cause the processor to deadlock.

Configurations affected

The erratum affects configurations that include the full FPU or the optimized FPU.

Conditions

The erratum can happen under the following conditions:

- A FPU Load or Store instruction is executed, but must be flushed after a branch mis-prediction or after an abort exception.
- Because this instruction is flushed, the instruction is discarded from the core pipeline, but still must be executed by the FPU pipeline to put the register interlock of the FPU in a known and valid state.
- Some FPU data-path operations might follow this FPU Load or Store instruction.
- Then the FPU is disabled by either setting FPEXC.EN bit to 0, or by setting CPACR.{cp10, cp11} to values different from {0b11, 0b11}.

Under certain unusual timing circumstances, because of the out-of-order capability of the processor, the FPU might be disabled before the flushed FPU Load or Store instruction is executed within the FPU, and this disable removes the clock that drives the FPU. Thus some FPU instructions are never executed.

Implications

The erratum might cause a processor deadlock.

Workaround

The recommended workaround requires setting bit [15] (Force data engine clock) in the undocumented CP15 Diagnosis register by the following read-modify-write sequence:

```
MRC     p15,0,r0,c15,c0,1   ; read undocumented CP15 Diagnosis register
ORR     r0,r0,#(1<<15)          ; set bit[15] of register r0 to 1
MCR     p15,0,r0,c15,c0,1   ; write undocumented CP15 Diagnosis register
```

794729: The QoS feature might cause data corruption when data coherency is enabled

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

This erratum applies to a Cortex-R7 MPCore processor that has two processors and has data coherency enabled. Each processor allocates data in cache for all accesses that are mapped to a Normal Cacheable memory region. When all ways of the data cache are valid for a particular index, a victim data way is chosen. The victim data is evicted and the new data is allocated instead.

When the QoS feature is enabled in the Auxiliary Control Register, the processor might, under certain conditions, return data to its register bank without allocating it in its data cache. In this case, the victim data is not evicted and remains in the data cache, unless the victim data is requested by the other processor at the same time. Because of this erratum, although the victim data is removed from the data cache of one processor, it remains visible from the Snoop Control Unit (SCU). In particular, a subsequent write access to the address of the victim data hits in the SCU when it should not, and this means the write access is made to a non-valid location. The required write access is lost, and the access to the non-valid location might cause data corruption.

Configurations affected

The erratum affects configurations that have two processors and ECC present, even if ECC is not enabled.

Conditions

The erratum can happen under the following conditions:

- Data coherency is enabled in both processors, by setting ACTLR.SMP bit to 1.
- QoS is enabled, at least in one processor, by setting ACTLR.QoS to 1. This description assumes this bit is set in processor 0.
- Processor 0 executes a store to address A1 that requires a linefill in its data cache. This linefill also requires an eviction of data at address A2. A2 and A1 are in the same cache line. Under certain circumstances of QoS operation, this linefill is abandoned. Note that a cache line is four doublewords.
- Processor 1 executes a store to address A2 that requires a linefill in its data cache. Processor 0 executes a store to address A2.

Under certain timing circumstances, the last store executed by processor 0 might be lost.

Implications

The erratum might cause data corruption.

Workaround

There is no workaround.

However the QoS feature is optional and configurable on a per processor basis by software. Disabling it may impact the real time performance of the processor.

794773: A short loop with DMB might prevent a CP15 broadcast operation making forward progress, causing a denial of service

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

In a Cortex-R7 configuration with two processors, one processor that continuously executes a short loop containing a DMB instruction might prevent a CP15 operation broadcast by the other processor making further progress, causing a denial of service.

Configurations affected

This erratum affects Cortex-R7 configurations with two processors.

Conditions

The erratum requires the following conditions:

- The two processors are working in SMP mode (ACTLR.SMP=1).
- One of the processors continuously executes a short loop containing at least one DMB instruction.
- The other processor executes a CP15 maintenance operation that is broadcast. This processor must have the broadcasting of CP15 operations enabled (ACTLR.FW=1).

For the erratum to occur, the short loop containing the DMB instruction must meet all of the following additional conditions:

- No more than 10 instructions other than the DMB are executed between each DMB.
- No non-conditional Load or Store, or conditional Load or Store which pass the condition code check, are executed between each DMB.

When all the conditions for the erratum are met, the short loop creates a continuous stream of DMB instructions. This might cause a denial of service, by preventing the processor executing the short loop from executing the received broadcast CP15 operation. As a result, the processor that originally executed the broadcast CP15 operation is stalled until the execution of the loop is interrupted.

Note that because the processor issuing the CP15 broadcast operation cannot complete operation, it might not be able to enter any debug-mode or to take any interrupt. If the processor executing the short loop also cannot be interrupted, for example if it has disabled its interrupts, or if no interrupts are routed to this processor, this erratum might cause a system livelock.

Implications

The erratum might create performance issues, or in the worst case it might cause a system livelock if the processor executing the DMB is in an infinite loop that cannot be interrupted.

Workaround

This erratum can be worked round by setting bit[25] of the undocumented Diagnostic Control Register to 1. This register is encoded as CP15 c15 0 c0 1. This bit must be written with a Read/Modify/Write code sequence. When it is set, this bit causes the DMB instruction to be decoded and executed like a DSB.

Using this software workaround is not expected to have any impact on the overall performance of the processor on a typical code base.

Other workarounds are also available for this erratum, to either prevent or interrupt the continuous stream of DMB instructions that cause the deadlock. For example:

- Inserting a non-conditional Load or Store instruction in the loop between each DMB.
- Inserting additional instructions in the loop, such as NOPs, to avoid the processor seeing back-to-back DMB instructions.
- Making the processor that executes the short loop take regular interrupts.

794971: Data values might not be traced

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

When data trace is enabled in the ETM, data values and data addresses are normally traced as they occur.

Because of this erratum, when several load or store operations executed in close proximity are followed by a conditional instruction that fails its condition code check, some of the data values might not be traced.

The erratum affects the trace only of data values. The address for the transfer is traced correctly. The instruction trace stream is not affected. Other data address and value pairs are not affected and are completely decompressible.

Resources might be affected if they are sensitive to the data value that is not traced.

Configurations affected

This erratum affects all configurations that include and are using the ETM.

Conditions

The erratum can occur under the following conditions:

- Data value tracing is enabled in the ETM, by setting TRCCONFIGR.DV to 1.
- A sequence of several load or store instructions completes in close proximity.
- A conditional instruction which fails its condition code check also executes in close proximity.

Implications

Individual data values might be dropped from the data trace. As a consequence, a decompressor might process the trace as if the transfer never completed.

Workaround

There is no workaround for this erratum.

854324: Write access to the SCU Control Register may result in unexpected value

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

The SCU Control Register allows to enable various features, using bits [15:12] (ECC on the bus enable bits) and [6:0] (other control bits). Others bits are SBZ. In this erratum, bits[15:12] (byte1) are using the byte write strobe of byte0. Therefore, a byte write access might result in an unexpected value.

Configurations affected

This erratum affects all configurations of the processor that use ECC on the bus.

Conditions

The erratum can happen if one of the following conditions happens:

- A STRB <Rn>, [SCU_CONTROL_REG] is executed and will overwrite the ECC enable bits with unknown value or
- A STRB <Rn>, [SCU_CONTROL_REG+1] is executed and will not update the ECC enable bits as expected.

Implications

Because of this erratum, the ECC on the bus might be unintentionally disabled/enabled or might simply not be disabled/enabled as expected.

Workaround

The recommended workaround is to use a STR instruction in a read-modify-write sequence to write in the SCU Control Register.

```
        LDR r0,[SCU_CONTROL_REG]
<modify r0>
        STR r0,[SCU_CONTROL_REG]
```

854436: CP15 write access to Cache Tag RAM location might lead to data corruption

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

The Cache and TCM Debug Operation Register (CTDOR) allows to select either the Cache or the TCM and the read or write operation to perform. Then, a subsequent access to the RAM Access Data Registers (RADRLO and RADRHI) allows to set the value to write to the selected RAM or store the returned value of the selected RAM.

Because of this erratum, when the CTDOR selects a write operation to the Tag RAM of the Data Cache, the store buffer merging capability of the processor might unintentionally merge some date to write, which might lead to data corruption.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The erratum requires the following conditions:

- The Cache and TCM Debug Operation Register (CTDOR) selects the Tag RAM of the Data Cache and selects a write operation.
- A write is done to the RAM Access Data Register (RADRLO).

Implications

Because of this erratum, the processor might lead to data corruption in the Data Cache.

For configurations that use ECC, it is not possible to analyze an erroneous RAM location in the Tag RAM of the Data Cache and determine whether an ECC error is a soft error or a hard error, as recommended in the basic ECC scheme described in the TRM.

Workaround

In this erratum, a DSB is not sufficient to drain the store buffer correctly and reset the merging capabilities of its slots. The recommended workaround is to perform a Clean Data Cache by set/way on any cache line before executing the CP15 write access to the Tag RAM of the Data Cache by using the following sequence:

<Clean Data Cache by Set/Way> by using the same index/way for instance of the CP15 write access to the Tag RAM of the Data Cache to be executed

<CP15 write access to the Tag RAM of the Data Cache>

854437: In Low Latency Interrupt mode, an IRQ or a FIQ received might generate a data abort

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

In Low Latency Interrupt mode, an IRQ or a FIQ received has a mechanism to flush the pipeline of the processor faster than in normal mode. Because of this erratum, an IRQ or a FIQ received might generate a data abort.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The erratum requires the following conditions:

- Low Latency Interrupt mode is enabled by setting SCTLR.FI to 1.
- Several consecutive IRQ or FIQ occur.

In this case, and under certain timing circumstances, an abort might be taken.

Implications

Because of this erratum, the processor might take the wrong exception handler and not treat the expected IRQ or FIQ.

Workaround

The recommended workaround requires setting bit [21] (Disable of Load/Store flush in Low Latency Interrupt mode) in the undocumented CP15 Diagnosis register by the following read-modify-write sequence:

```
MRC     p15,0,r0,c15,c0,1   ; read undocumented CP15 Diagnosis register
ORR     r0,r0,#(1<<21)
MCR     p15,0,r0,c15,c0,1   ; write undocumented CP15 Diagnosis register
```


854722: Broadcasting invalidate instruction Cache by MVA may stall a core

Category B

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

Whenever a core executes a broadcasting invalidate Instruction Cache by MVA operation to another core, for example when copying executable code from flash to memory, the other cores flush and restart their prefetch unit each time without progressing. Because of this erratum, the other cores might stop their execution until there is no remaining broadcasting operation occurring anymore.

Configurations affected

This erratum affects all configurations of the processor with two or more cores.

Conditions

The erratum requires the following conditions:

- Two or more cores are working in SMP mode (by setting ACTLR.SMP to 1), and have the Cache maintenance broadcast enabled (by setting ACTLR.FW to 1).
- A core runs several successive invalidate Instruction Cache by MVA.
- The other cores, flushing and restarting their prefetch unit, need to execute some code that is either non-cacheable or cacheable, but that is neither in the Instruction Cache nor in the ITCM at this moment.

In this case, under certain timing circumstances, the other cores are not able to fetch the code they need to execute, and this might stop their execution.

Implications

Because of this erratum, if the interrupt handler code is either non-cacheable or cacheable, but is neither in the Instruction Cache nor in the ITCM at this moment, the other cores are not able to fetch it and, therefore, execute it. This increases the interrupt latency time until there is no longer any remaining invalidate Instruction Cache by MVA operation.

Workaround

A workaround for this erratum is to put critical interrupt code in the ITCM to have a reasonable interrupt latency.

2.6. Category B (Rare)

2.7. Category C

786371: Accesses to a Normal Memory Cacheable region going to the AXI peripheral port might deadlock the processor

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

The AXI peripheral port is memory-mapped. The start and end address for the peripheral port are defined by processor input pins. Memory accesses also have some memory attributes defined in the MPU. For consistency, accesses to the AXI peripheral port must have their memory attributes set to Strongly-ordered or Device. Other memory attributes, for example Normal Write-Back, Write-Allocate cacheable or Normal Non-cacheable generate an imprecise abort. This is the expected memory system behavior. Because of this erratum, the access might deadlock if the memory attributes are cacheable.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum requires no conditions.

Implications

The erratum might cause the processor to deadlock. Note that accesses to the AXI peripheral port must have their attributes set to Strongly-ordered or Device, so if this rule is followed, this erratum should never occur.

Workaround

There is no work-around.

787175: Data values might be missing from data trace when dynamic clock gating is enabled

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

When data value trace is enabled in the ETM, data addresses and data values are traced as they occur. When dynamic clock gating of the processor is enabled, the core clk can be dynamically stopped when the core pipeline is empty and the instruction cache is waiting for a linefill. Under certain conditions, the core clk might be stopped temporarily, meaning a data value element is not traced. This means a data address might not have a corresponding data value. The instruction trace stream is not affected. Other data address and data value pairs are not affected and completely decompressable.

Configurations affected

The erratum affects all configurations that include and are using the ETM.

Conditions

The erratum can happen under the following conditions:

- Data value tracing is enabled in the ETM, by setting TRCCONFIGR.DV to 1.
- Dynamic clock gating is enabled in the processor, by setting bit 0 of CP15 Power Control Register (p15,0,<Rd>,c15,c0,0).
- An unaligned store is traced that crosses a word boundary.
- The core pipeline is empty and the instruction cache is waiting a line fill.

In this case, under certain timing circumstances, the core clk is temporarily stopped and a data value might be missing from the data trace.

Implications

Individual data values might be dropped from the data trace. A decompressor will probably treat this as if a transfer never completed.

Workaround

The recommended workaround requires setting bit [5] (Disable instruction cache line fill wait bit) in the undocumented CP15 Diagnosis register by the following read-modify-write sequence:

```
MRC     p15,0,r0,c15,c0,2   ; read undocumented CP15 Diagnosis register
ORR     r0,r0,#(1<<5)
MCR     p15,0,r0,c15,c0,2   ; write undocumented CP15 Diagnosis register
```

By doing this, the core clk is not stopped and the erratum cannot occur. As the condition to dynamically stop the core clk is rare, the impact of this workaround has negligible impact on power consumption.

793269: “Write Context ID” event is updated on read access

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0

Description

When selected, the “Write Context ID” event (0x0B) of the Performance Monitoring Unit (PMU) increments a counter whenever an instruction that writes to the Context ID register, CONTEXTIDR, is architecturally executed. However this erratum means that an instruction that reads the Context ID register also updates this counter.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum can happen under the following conditions:

- A PMU counter is enabled, by setting PMCNTENSET.Px bit to 1. x identifies a single event counter, and takes a value from 0 to 7.
- The “Write Context ID” event is mapped to this selected PMU counter:
  - The chosen PMU counter is selected, by setting PMSELR.SEL to x (same value as above).
  - The “Write Context ID” event is mapped to this selected PMU, by setting PMXEVTYPER.evtCount to 0x0B.
- The PMU is enabled, by setting the PMCR.E bit to 1.
- A read access occurs to the CONTEXTIDR.

In this situation the PMU updates the counter when it should not.

Implications

The erratum alters the accuracy of the “Write Context ID” event.

Workaround

There is no workaround for this erratum.

794719: Performance monitor counters continue counting after being disabled

Category C
Products Affected: Cortex-R7 CPU.
Present in: r0p0

Description

The counter of the Performance Monitoring Unit (PMU) can be enabled or disabled dynamically.

However, this erratum means that, if a counter has been enabled, whatever event is selected,

the counter cannot be disabled and continues to count forever when the selected event occurs.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum can happen under the following conditions:

- A PMU counter is enabled, by setting a PMCNTENSET.Px bit to 1. x identifies a single event counter, and takes a value from 0 to 7.
- The event “y” is mapped to this selected PMU counter, as follows:
  - The chosen PMU counter is selected, by setting PMSELR.SEL to x (the same value as above).
  - The “y” event is mapped to this selected PMU, by setting PMXEVTYPER.evtCount to the encoding for “y”.
- The PMU is enabled, by setting the PMCR.E bit to 1.
- The PMU is disabled, by setting the PMCR.E bit to 0 or by setting PMCNTENCLR.Px bit to 1.

In this situation, whenever the event “y” occurs, the PMU updates the counter although it should not do so.

Implications

The erratum might cause increments to the event counter when not needed.

Workaround

There is no workaround.

800669: DBGPRSR Sticky Reset status bit is set to 1 by the CPU debug reset instead of by the CPU non-debug reset

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

DBGPRSR.SR, bit [3], is the Sticky Reset status bit. The ARM architecture specifies that the processor sets this bit to 1 when the non-debug logic of the processor is in reset state. Because of this erratum, the Cortex-R7 sets this bit to 1 when the debug logic of the processor is in reset state, instead of when the non-debug logic of the processor is in reset state.

Configurations affected

The erratum affects all configurations.

Conditions

The erratum requires no conditions.

Implications

Because of the erratum:

- DBGPRSR.SR might not be set to 1 when it should, when the non-debug logic of the processor is in reset state.
- DBGPRSR.SR might be set to 1 when it should not, when the debug logic of the processor is in reset state.

In both cases, the DBGPRSR.SR bit value might be corrupted, which might prevent the debug logic from correctly detecting when the non-debug logic of the processor has been reset.

Workaround

There is no work-around.

854271: CP15 read access to ITCM RAM location might return incorrect data value

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

The Cache and TCM Debug Operation Register (CTDOR) allows to select either the Cache or the TCM. Then, subsequent accesses to the RAM Access Data Registers (RADRLO and RADRHI) allow to get to the returned value of the RAM.

Because of this erratum, when the CTDOR selects the ITCM, the processor might not read the ITCM RAM location as expected, but might read an incorrect data value instead.

Configurations affected

This erratum affects all configurations of the processor that use ECC.

Conditions

The erratum requires the following conditions:

- The ECC is enabled on ITCM (because bit [10] of Auxiliary Control Register, ACTLR, is set to 1).
- The ITCMEER register contains a valid information.
- The CP15 read access hits the ITCMEER register.

Implications

Because of this erratum, it is not possible to analyze an erroneous RAM location in ITCM and determine whether an ECC error is a soft-error or a hard-error, as recommended in the basic ECC scheme described in the TRM.

Workaround

The recommended workaround for this erratum is to save the ITCMEER register and then clear it before analyzing an erroneous RAM location by using the following sequence:

```
MRC p15, 0, r0, c15, c5, 0; Read ITCM ECC entry
<save r0>  (save ITCMEER contents)
BIC r0,r0,#0x1; Clear the valid bit of the ITCM ECC entry
MCR p15, 0, r0, c15, c5, 0; Write ITCM ECC entry
<Analyze the erroneous RAM location in ITCM>
<restore r0> (restore ITCMEER contents)
MCR p15, 0, r0, c15, c5, 0; Write ITCM ECC entry
```

854272: UNDEFINED instructions might not raise an UNDEF exception

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

Executing some UNDEFINED opcodes might not raise an UNDEF exception, and the processor will either execute an SDIV instruction or a NOP instruction.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The following opcode is affected:

Thumb2: 1111 1011 1001 Rn (1)(1)(1)(1) Rd 0001 Rm

Implications

If the opcode is of the form:

Thumb2: 1111 1011 1001 Rn (1)(1)(1)(1) Rd 0001 Rm

then an SDIV instruction will be executed.

Otherwise, a NOP instruction will be executed.

Workaround

There is no workaround. However, ARM does not guarantee that this encoding will raise an UNDEF exception. This means the software must not rely on taking an UNDEF trap when it executes an opcode within that encoding space.

854771: Eviction request missing in tag RAM can incorrectly detect an ECC error in Data RAM

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

When an ECC error occurs on the Data RAM of the Data Cache, it is notified to the system and then an automatic cache maintenance operation is performed.

This maintenance operation will write the Tag RAM of the Data Cache, to invalidate it, but doesn't write the Data RAM.

If a following eviction is requested on the same index and way, it will read both Tag and Data RAMs, and report an ECC error from the Data part, even if it is missing in Tag ram. This will lead to the same error reported another time in the error bank, and on the external pin RAMERR.

Configurations affected

This erratum affects all configurations of the processor that use ECC.

Conditions

The erratum requires the following conditions:

- The ECC is enabled on caches (because bit [9] of Auxiliary Control Register, ACTLR, is set to 1).
- An ECC error is seen on Data RAM of the Data Cache.
- One of two following conditions:
  - Error bank was already full on the first ECC error.
  - or, error bank entry is cleared by software after this first ECC error.
- An eviction is requested on the same index/way than the previous ECC error and detects a new ECC error. This eviction can be a natural eviction from a linefill, or requested from a cp15 clean invalidate Data Cache by MVA operation.

Implications

Because of this erratum, it might be possible to report an ECC error twice to the system.

Workaround

On the clearing of the error bank entry, software can write to the Data RAM and the ECC Data RAM with valid values, to remove any ECC error still present.

1414441: The debugger might hang in Lock mode if it accesses CPU1 registers

Category C

Products Affected: Cortex-R7 CPU.

Present in: r0p0, r0p1

Description

In Split/Lock configuration and when **SAFEMODE** pin is asserted HIGH (Lock mode), CPU1 is locked to CPU0 and they both behave as a single CPU.

However, as the ROM table always indicates the presence of 2 CPUs in the cluster, the debugger might attempt to access CPU1 registers through APB.

In this case, the access will never complete and the debugger will hang.

Configurations affected

This erratum affects the Split/Lock configuration, in Lock mode.

Conditions

1. The cluster is built with the Split/Lock configuration and runs in Lock mode.
2. The debugger performs an access to any CPU1 register accessible through the APB bus.

If the above conditions are met, then **PREADYDBG** is driven LOW which prevents the access to complete.

Implications

This erratum might cause the debugger to hang.

Workaround

There is no workaround to this erratum. The debugger must avoid accessing CPU1 registers in this case.