ARM Processor Cortex™-R4 and Cortex-R4F

Product Revision r1

Software Developers Errata Notice

Non-Confidential - Released

ARM-EPM-018901 v2.2 Released

Software Developers Errata Notice

Non-Confidential Proprietary Notice

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.

20 Jun 2012: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
16	New	780124	CatB	Rare	Processor might deadlock or lose data when configured with cache-ECC

02 May 2012: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
10	New	447975	CatA	Rare	Long TCM wait-states may cause an ECC correction to livelock
11	New	450316	CatB		Two-register store multiple with writeback to device memory can be interrupted
12	New	534616	CatB		Thumb MOVT, BFI and BFC can give wrong result when dual-issued with floating point VMOV
13	New	577077	CatB		Thumb STREXD treated as NOP if same register used for both source operands
14	New	457929	CatB	Rare	Data cache errors may not cause an event to be signaled
15	New	737195	CatB	Rare	Conditional VMRS APSR_nzcv, FPSCR may evaluate with incorrect flags
18	New	412027	CatC		Debug reset does not reset DBGDSCR when in Standby Mode
19	New	436726	CatC		Instruction Set Attribute Register 4 incorrect
20	New	441181	CatC		Debug halt request causes correctable error to generate abort
21	New	442112	CatC		Instruction Set Attribute Register 1 incorrect
22	New	451324	CatC		Instruction fetch can generate spurious device read
23	New	452032	CatC		Processor can deadlock when debug mode enables cleared
24	New	453266	CatC		Spurious error event may be generated on an instruction cache invalidate
25	New	454166	CatC		TCM error events may be incorrectly signaled
26	New	459904	CatC		PMCR IDCODE field is incorrect
27	New	461513	CatC		Memory Model Feature Register 3 value is incorrect
28	New	463964	CatC		Access to TCM with wait cycles via AXI slave interface may livelock.
29	New	500166	CatC		On exit from Dormant Mode, data marked as dirty can be lost
30	New	599517	CatC		CP15 Auxiliary ID and Prefetch Instruction accesses are UNDEFINED
31	New	639819	CatC		An instruction which causes a data watchpoint to match is incorrectly traced
32	New	685018	CatC		Processor exits debug halt state when DBGEN is low
33	New	720270	CatC		Latched DTR-full flags not updated correctly on DTR access
34	New	722412	CatC		CPACR.ASEDIS and CPACR.D32DIS incorrect when configured with floating-point
35	New	726554	CatC		DBGDSCR.ADAdiscard is wrong when DBGDSCR.DBGack set
36	New	732313	CatC		DBGPRSR sticky reset status may not be set on CPU reset
37	New	733066	CatC		CFLR reports correctable fault location inconsistently when multiple correctable faults occur simultaneously
38	New	736960	CatC		Debug halt exceptions are always shown as cancelling on ETM interface
39	New	745322	CatC		DBGDSCR.SPNIDdis and DBGDSCR.SPIDdis are always zero
40	New	748619	CatC		Missing reset exception on ETM interface
41	New	754269	CatC		Register corruption during a load-multiple instruction at an exception vector
42	New	756022	CatC		Watchpointed access in a store-multiple is not masked
43	New	758119	CatC		Incorrect read data may be returned for a load to the D-cache
44	New	758269	CatC		Watchpoint on a load or store multiple may be missed

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-R4 and Cortex-R4F processors.

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 r1 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	r1p0	r1p1	r1p2	r1p3	r1p4
447975	CatA	Rare	Long TCM wait-states may cause an ECC correction to livelock	X				
577077	CatB		Thumb STREXD treated as NOP if same register used for both source operands	X	X	X	X	
534616	CatB		Thumb MOVT, BFI and BFC can give wrong result when dual-issued with floating point VMOV	X	X	X		
450316	CatB		Two-register store multiple with writeback to device memory can be interrupted	X				
780124	CatB	Rare	Processor might deadlock or lose data when configured with cache-ECC	X	X	X	X	X
737195	CatB	Rare	Conditional VMRS APSR_nzcv, FPSCR may evaluate with incorrect flags	X	X	X	X	
457929	CatB	Rare	Data cache errors may not cause an event to be signaled		X			
758269	CatC		Watchpoint on a load or store multiple may be missed	X	X	X	X	
758119	CatC		Incorrect read data may be returned for a load to the D-cache	X	X	X	X	
756022	CatC		Watchpointed access in a store-multiple is not masked	X	X	X	X	
754269	CatC		Register corruption during a load-multiple instruction at an exception vector	X	X	X	X	
748619	CatC		Missing reset exception on ETM interface	X	X	X	X	
745322	CatC		DBGDSCR.SPNIDdis and DBGDSCR.SPIDdis are always zero	X	X	X	X	
736960	CatC		Debug halt exceptions are always shown as cancelling on ETM interface	X	X	X	X	
733066	CatC		CFLR reports correctable fault location inconsistently when multiple correctable faults occur simultaneously	X	X	X	X	

ID	Cat	Rare	Summary of Erratum	r1p0	r1p1	r1p2	r1p3	r1p4
732313	CatC		DBGPRSR sticky reset status may not be set on CPU reset	X	X	X	X	
726554	CatC		DBGDSCR.ADAdiscard is wrong when DBGDSCR.DBGack set	X	X	X	X	
722412	CatC		CPACR.ASEDIS and CPACR.D32DIS incorrect when configured with floating-point	X	X	X	X	
720270	CatC		Latched DTR-full flags not updated correctly on DTR access	X	X	X	X	
685018	CatC		Processor exits debug halt state when DBGEN is low	X	X	X	X	
639819	CatC		An instruction which causes a data watchpoint to match is incorrectly traced	X	X	X	X	
599517	CatC		CP15 Auxiliary ID and Prefetch Instruction accesses are UNDEFINED	X	X	X	X	
500166	CatC		On exit from Dormant Mode, data marked as dirty can be lost	X	X	X		
463964	CatC		Access to TCM with wait cycles via AXI slave interface may livelock.	X				
461513	CatC		Memory Model Feature Register 3 value is incorrect	X	X	X		
459904	CatC		PMCR IDCODE field is incorrect	X				
454166	CatC		TCM error events may be incorrectly signaled	X	X			
453266	CatC		Spurious error event may be generated on an instruction cache invalidate		X			
452032	CatC		Processor can deadlock when debug mode enables cleared	X	X	X	X	
451324	CatC		Instruction fetch can generate spurious device read	X				
442112	CatC		Instruction Set Attribute Register 1 incorrect	X				
441181	CatC		Debug halt request causes correctable error to generate abort	X				
436726	CatC		Instruction Set Attribute Register 4 incorrect	X				
412027	CatC		Debug reset does not reset DBGDSCR when in Standby Mode	X	X	X	X	

2.3. Category A

There are no errata in this category

2.4. Category A (Rare)

447975: Long TCM wait-states may cause an ECC correction to livelock

Category A Rare
Products Affected: Cortex-R4.
Present in: r1p0

Description

In revision r1p0 and later, the Cortex-R4 TCM interfaces can be configured to include ECC logic, with error correction being performed on either 32-bit or 64-bit aligned chunks of data.

If a TCM ECC scheme is enabled and a write to that TCM writes part, but not all of a data chunk (for example an STRH to a TCM with 32-bit ECC), the processor must first read the whole of that data chunk before it can compute the new error code to write to the TCM.

Read-modify-write can also be explicitly enabled using bits [1:0] of the second auxiliary control register. If this is enabled, then the processor behaves as if a 64-bit ECC scheme were enabled.

The *TCWAIT signals on the TCM ports can be asserted, in the data phase of an access, to stall that access and cancel any following access that is in its address phase. The TCM controller might be built to assert the wait signal for a number of cycles for every access, so that every access takes a fixed number of cycles.

If a TCM access is always given the same number of wait cycles, and each wait is six or more cycles, then due to this erratum, it is possible for the correction of an ECC error to livelock the processor.

Conditions

1) ECC support is enabled on the TCM, and
2) The TCM has fixed wait states of 6 or more cycles, and
3) A store to TCM is executed, and
4) A load, or a store that requires a read-modify-write, is executed, and
5) The second access gets a correctable ECC error on the data it reads from the TCM, and
6) Bits [31:3] of the address of both accesses must be the same, and the second access must read some or all of the bytes being written by the first access.

Implications

If this erratum is triggered, the processor will get in to a livelock as it continually tries to reexecute the same instruction. It will continue in the livelock until an exception, such as an interrupt, imprecise abort, or debug request, alters the instruction flow and allows the first store to complete successfully.

If the TCM does not need any wait states, or the number of wait states is less than 6 cycles, then this erratum cannot occur.

Workaround

No workaround is available.

2.5. Category B

450316: Two-register store multiple with writeback to device memory can be interrupted

Category B

Products Affected: Cortex-R4.

Present in: r1p0

Description

Instructions that generate memory accesses to locations in regions marked as "device" or "strongly-ordered" must be atomic. An instruction that implies a number of memory accesses must either fully complete, or perform none of the accesses.

Due to this erratum, certain types of store instruction can be interrupted after performing memory writes to device or strongly-ordered memory, but before they are complete. If the interrupt handler returns to the program in the normal way, the instruction will be repeated, and therefore some memory writes will be performed twice.

Conditions

This erratum will occur if one of the following types of instruction is interrupted after it has started execution, and the memory being accessed by that instruction is in a region marked as device or strongly-ordered by the MPU.

1) STM or FSTMS of two registers with base register update, to an address that is doubleword-aligned, or
2) FSTMD of one register with base register update, to an address that is doubleword-aligned, or
3) FSTD with base register update to an address that is doubleword-aligned.

Implications

Device, or strongly-ordered type memory is usually used for peripherals which are sensitive to the number of memory accesses performed. If this erratum occurs, then more writes than required will be performed, and, depending on the nature of the peripheral, data corruption can occur.

Workaround

This erratum can be avoided by either:

1) Avoiding the instructions listed in the conditions section. For example, STMIA r0!,{r1,r2} could be replaced with: STRD r1,[r0], #8 for the same behavior. If the registers to be transferred are not adjacent, two instructions could be used. For example, STMIA r0!,{r1,r3} could be replaced with STMIA r0,{r1,r3}, ADD r0,r0,#8, which would take the same number of cycles to execute provided the base address was doubleword-aligned. Or
2) Setting bit [6] of the auxiliary control register to disable low-interrupt-latency behaviour on all load/store instructions. This has the side-effect of increasing the worst-case interrupt latency, making it comparable to the ARM9 family for a similar system configuration.

534616: Thumb MOVT, BFI and BFC can give wrong result when dual-issued with floating point VMOV

Category B

Products Affected: Cortex-R4F.

Present in: r1p0, r1p1, r1p2

Description

The Cortex-R4F processor can, in some cases, issue and execute two instructions simultaneously for improved throughput. This is known as dual-issuing.

Because of this erratum, some BFC, BFI and MOVT instructions will give an incorrect result when dual-issued with a preceding VMOV instruction which transfers 32-bits of data from the VFP register file to an ARM core register.

Conditions

1) The Cortex-R4 processor is implemented with floating point logic, ie. Cortex-R4F, and
2) The VFP logic (cp10 and cp11) is enabled in the Coprocessor Access Register for the current privilege level, and
3) The VFP logic is enabled by means of the EN bit in the FPEXC register, and
4) The processor is in Thumb state, and
5) The program contains a VMOV instruction which transfers either one single-precision register or half of a double-precision register from the VFP register file to the ARM core register file, and
6) Immediately following the VMOV instruction is a MOVT, BFI or BFC instruction, and
7) The destination register, Rd, for the VMOV instruction is the same as the destination register, Rd, for the MOVT, BFI or BFC instruction, and
8) When the VMOV instruction is being decoded, the second instruction has also been fetched so that it can be decoded and issued simultaneously, and
9) Dual-issue case F3 is enabled, that is, bit [18] of the Secondary Auxiliary Control Register is 0.

Implications

If the erratum occurs, the result of the MOVT, BFI or BFC instruction is incorrect. The result is computed as if the VMOV instruction had not been executed.

Workaround

This erratum can be avoided by disabling case F3 dual-issuing. Set bit [18] of the Secondary Auxiliary Control Register. Note: this workaround will have a small impact on the instruction throughput of the processor. The impact will depend on what code is being executed, but ARM has not been able to measure any noticeable impact during internal trials.

577077: Thumb STREXD treated as NOP if same register used for both source operands

Category B

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The ARM Architecture permits the Thumb STREXD instruction to be encoded with the same register used for both transfer registers (Rt and Rt2). Because of this erratum, the Cortex-R4 processor treats such an encoding as UNPREDICTABLE and executes it as a NOP.

Conditions

This erratum occurs when the processor is in Thumb state and a STREXD instruction is executed which has Rt = Rt2.

Note: this instruction was new in ARM Architecture version 7 (ARMv7). It is not present in ARMv6T2 or other earlier architecture versions.

Implications

If this erratum occurs the destination register, Rd, which indicates the status of the instruction, is not updated and no memory transaction takes place. If the software is attempting to perform an exclusive read-modify-write sequence, then it might either incorrectly complete without memory being written, or loop forever attempting to complete the sequence.

Workaround

The erratum can be avoided by using two different registers for the data to be transferred by a STREXD instruction. This may involve copying the data in the transfer register to a second, different register for use by the STREXD.

2.6. Category B (Rare)

457929: Data cache errors may not cause an event to be signaled

Category B Rare
Products Affected: Cortex-R4.
Present in: r1p1

Description

The Cortex-R4 processor contains a Performance Monitoring Unit that can be configured to count various events that happen, and an event bus interface to export these events to other components on the SoC.

Event numbers 0x4C, 0x4D, 0x60, and 0x61 count the number of errors detected on accesses to the data cache. The events are separated into recoverable and unrecoverable errors, and in which RAM the error was detected. A detected error should be signaled on one or more of the events, however in certain situations an error that occurred may not get signaled on any of these events.

Conditions

1) The data cache supports parity or ECC protection, and error checking is enabled in the auxiliary control register.
2) The processor performs a load instruction to normal cacheable memory.
3) A parity or ECC error is detected in the tag or data RAMs accessed during the load.
4) The cycle after the load, a second load instruction is executed.
5) This second load causes the pipeline to stall. This could be because of any of the following:
    - the load is to TCM, and must wait for an instruction fetch or AXI slave access to the same TCM to finish.
    - the load is not to the TCM, and must wait for the parity or ECC error to be corrected in the cache before it can start.
    - the load instruction mispredicted whether to access the cache, ATCM, B0TCM or B1TCM.
    - the load instruction reads some or all of the same bytes as an earlier store instruction wrote, and that store instruction is still present in the store queue.

Implications

Some systems may use events to initiate some action when an error is detected, for example to cause an interrupt so that software can then later examine the error in more detail. If this erratum occurs then in such a system the error will go unnoticed. However any data aborts will still be generated correctly, and the Correctable Fault Location Register will be updated correctly.

Workaround

Software can alter bits [5:3] of the Auxiliary Control Register to enable aborts to be taken on all detected cache errors. Depending on how the external system uses the events, the abort handler may be able to be used to perform any necessary tasks when an error is detected. However the information as to whether it was the tag, dirty or data RAM containing the error is not available to the abort handler.

737195: Conditional VMRS APSR_nzcv, FPSCR may evaluate with incorrect flags

Category B Rare

Products Affected: Cortex-R4F.

Present in: r1p0, r1p1, r1p2, r1p3

Description

Under certain circumstances, a conditional `VMRS APSR_nzcv, FPSCR` instruction may evaluate its condition codes using the wrong flags and therefore incorrectly execute or not execute.

Conditions

The erratum requires the following sequence of instructions in ARM-state:

1) `VMRS<c> APSR_nzcv, FPSCR` (formerly `FMSTAT<c>`), where the condition (`<c>`) on the instruction is not 'always' (i.e. `<c>` is not `AL`. Note: the condition `AL` is the default and can also be omitted). This instruction immediately following:

2) A flag-setting integer multiply or multiply and accumulate instruction (e.g. `MULS`), which follows:

3) A single-precision floating-point multiply-accumulate (FP-MAC) instruction (e.g. `VMLA`), timed such that the accumulate operation is inserted into the pipeline in the cycle in which the `VMRS` instruction is first attempted to be issued.

To meet the above timing requirements, the `VMRS` instruction must be three pipeline stages behind the FP-MAC. Depending on the rate in which the instructions are fetched, interlocks within this sequence and dual-issuing, this can be up to three other instructions between this pair, plus the multiply.

Out-of-order completion of FP-MAC instructions must be enabled.

Implications

If this erratum occurs, the VMRS instruction will pass or fail its condition codes incorrectly, and this will appear in any trace produced by the ETM. This can in turn corrupt the N, Z, C, V flag values in the CPSR which will typically affect the program flow.

Workaround

This erratum can be avoided by disabling out-of-order single-precision floating-point multiply-accumulate (SP-MAC) instruction completion. Set DOOFMACS, bit [16] in the Secondary Auxiliary Control Register. This will have the side-effect of reducing the performance of SP-MAC operations, though the impact will depend on how these instructions are used in your code.

780124: Processor might deadlock or lose data when configured with cache-ECC

Category B Rare

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3, r1p4

Description

The Cortex-R4 processor contains a 4-entry store buffer which buffers, merges and forwards data before it is written to the cache or the L2 memory system using the AXI-master interface. Because of this erratum it is possible for the store buffer to enter a state in which no existing writes will proceed. The effect of this state is either:

- The pipeline backs-up preventing any instruction execution or
- If a specific sequence of accesses is performed, execution resumes but write data is lost.

Configurations Affected

This erratum only affects processors configured with ECC on the data-cache and with at least one accessible region of memory that has Inner Write-Back Cacheable and Non-shareable attributes.

Configurations

Either of the following sequences of conditions is required for this erratum to occur:

1) The processor is implemented with data-cache ECC, and cache-ECC is enabled, and
2) The processor accesses a memory location that misses in the L1 data cache and causes a cache line to be read and allocated and
3) The processor performs a write to a Write-Back Cacheable location that hits before and misses after the linefill in step [2]. This write performs its cache lookup in the cycle before the line is reallocated by [2] and
4) The processor subsequently performs a read and a write to the same cache line as the write in step [3]. This read and write can occur in either order. The write is to a different doubleword than the write in step [3].

or:

1) The processor is implemented with data-cache ECC, and cache-ECC is enabled, and
2) The processor reads a Write-Back Cacheable memory location that misses in the L1 data cache and causes a cache line to be read and allocated but does not detect any ECC errors and
3) The processor performs a write to the same cache line as the read in step [2]. When the address is looked up in the cache it appears to hit because of an ECC error in the tag-RAM and
4) The processor subsequently performs a further write to the same cache line as the read in step [2], but not the same doubleword as the write in step [3].
5) A subsequent speculative cache read also detects an ECC error. This read can be to the same cache set and therefore detect the same error, or a different set that requires a second ECC error for this condition to be met.

In addition, both sets of conditions require specific timing relationships between the two accesses and are therefore affected by the timing of transactions on the AXI bus and the status of other ongoing writes in the store buffer.

Deadlock will not occur if either of the above sequences is followed by:

1) A read that misses in the cache and causes a linefill and
2) Two or more writes to the same cache line as the read in [1]. The two writes must be to different doublewords but one can be to the same doubleword as the read in [1]. A single store instruction can generate two such writes if it is not naturally aligned.

If this happens, some of the write data might be lost. Also, a following Non-cacheable or Device write might be allocated into the cache.

Note: all numerical cross-references above refer to items within the same list as the reference.

Implications

If this erratum occurs the processor will either deadlock or lose data. When deadlocked, the processor can take an interrupt but data loss or deadlock will eventually occur in the handler code.

This erratum is categorized as rare partially based on empirical evidence from a large number of parts in the field. This issue has only been seen on one project and in that case the time to failure is long and variable.

Workaround

You can avoid this erratum by setting ACTLR.DBWR (bit [14]) to 1. This setting disables an optimization to the internal transfer of bursts of write data to Normal memory. This setting also disables generation of AXI bursts by the processor for Write-Through and Non-cacheable Normal memory, but not Write-Back memory. Setting this bit to 1 might reduce the performance of writes to Normal memory by the processor. In benchmarking, the average performance reduction is less than 1% but routines that perform large block writes such as memset or memcpy show substantially more significant impacts. The impact of the workaround on memset and memcpy is strongly dependent on the performance and characteristics of the L2 memory system and on the exact instruction sequence used.

If your application permits it you can also avoid this erratum by disabling cache-ECC. Set ACTLR.CEC (bits [5:3]) to b100. This workaround does not reduce the performance of the processor, but disabling ECC has implications for reliability.

2.7. Category C

412027: Debug reset does not reset DBGDSCR when in Standby Mode

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The debug reset input, PRESETDBGn, resets the processor's debug registers as specified in the ARMv7R Architecture. Because of this erratum, when the processor is in Standby Mode and the clock has been gated off, PRESETDBGn fails to reset the Debug Status and Control Register (DBGDSCR).

Conditions

1) The DBGDSCR register has been written so that its contents differ from the reset values (most fields in this register reset to zero, though a few are UNKNOWN at reset), and

2) The processor is in Standby Mode, and the clocks have been gated off, that is STANDBYWFI is asserted, and

3) The debug reset, PRESETDBGn, is asserted and deasserted while the processor clocks remain gated off.

Note: the debug reset is commonly used to set the debug registers to a known state when a debugger is attached to the target processor.

Implications

If this erratum occurs, then after the reset the DBGDSCR register contains the values that it had before reset rather than the reset values. If the debugger relies on the reset values, then it may cause erroneous debug of the processor. For example, the DBGDSCR contains the ExtDCCmode field which controls the Data Communications Channel (DCC) access mode. If this field was previously set to Fast mode but the debugger assumes that it is in Non-blocking mode (the reset value) then debugger accesses to the DCC will cause the processor to execute instructions which were not expected.

Workaround

This erratum can be avoided by a workaround in the debug control software. Whenever the debugger (or other software) generates a debug reset, follow this with a write of zero to the DBGDSCR which forces all the fields to their reset values.

436726: Instruction Set Attribute Register 4 incorrect

Category C

Products Affected: Cortex-R4.

Present in: r1p0

Description

The value in the SynchPrim_instrs_frac field, bits [23:20] in Instruction Set Attributes Register 4 (ISAR4) is 0x3. This value is not supported by the ARM Architecture, and should be 0x0 instead.

Implications

Software that tries to identify the synchronization primitive instructions available on the processor will correctly determine which instructions Cortex-R4 provides, because these are indicated by the SynchPrim_instrs field, bits[15:12] in the Instruction Set Attributes Register 3 (ISAR3). However, the incorrect value in ISAR4 is Reserved in the ARM Architecture, so if this value is used in future revisions of the ARM Architecture, the erratum may cause software to identify instructions available that Cortex-R4 does not actually provide.

Workaround

There is no workaround for this erratum.

441181: Debug halt request causes correctable error to generate abort

Category C

Products Affected: Cortex-R4.

Present in: r1p0

Description

On the Cortex-R4 processor, single-bit ECC errors in the data returned from a TCM interface read can either:

1) Cause the processor to take an abort, if ECC correction is disabled in the secondary auxiliary control register (ATCMECC, bit [2] or BTCMECC, bit [3] set, according the TCM interface in use), or

2) Cause the processor to correct the data, store it back to the TCM and replay the instruction that found the error, if ECC correction is enabled in the secondary auxiliary control register.

This erratum means that under certain conditions the processor takes a data abort (case 1 above) instead of replaying (case 2). The erroneous data is never incorrectly used; the processor takes a data abort when it would have been expected to do a correction and replay.

Conditions

This erratum can occur when all of the following are true:

1) A single bit ECC error is detected on a TCM read, and

2) The processor is configured to correct the error and not abort, and

3) At the same time there is a pending debug halt request.

Note that the debug halt request must be caused by either asserting the debug request pin of the processor or writing to the debug run control register (DRCR), not by a breakpoint or a watchpoint.

Implications

The erratum can occur when a debugger attempts to halt the Cortex-R4 processor. If a data abort is being handled then state associated with that data abort can become corrupted and it becomes impossible to determine the cause or location of the abort.

Workaround

Following a debug halt request the system may become corrupted if the processor is in a state for which a data abort is fatal. For example, if the processor is executing a data abort handler and the link register and SPSR have not yet been stacked. This does not have a workaround, but it should be noted that the debug halt request is probably at the request of a user interacting with a debugger so it may be a rare and therefore acceptable side effect.

442112: Instruction Set Attribute Register 1 incorrect

Category C

Products Affected: Cortex-R4.

Present in: r1p0

Description

The value in the Interwork_instrs field, bits [27:24] of the Instruction Set Attribute Register 1 (ISAR1) is 0x2, but should be 0x3.

Implications

Software that attempts to determine the behavior of an ARM data-processing instruction with the PC as the destination, the S bit clear, and bit [1] or bit [0] of the result set will conclude that the behavior of Cortex-R4 is UNPREDICTABLE due to this erratum. In fact, such an instruction has BX-like behavior on Cortex-R4. This will not result in any erroneous behavior.

Workaround

There is no workaround for this erratum.

451324: Instruction fetch can generate spurious device read

Category C

Products Affected: Cortex-R4.

Present in: r1p0

Description

Memory within regions marked in the Memory Protection Unit (MPU) as "device" or "strongly-ordered" type must only be accessed as explicitly coded in the program. For example, an LDRH instruction to load a half word from such a location must produce one, and only one 16-bit access at that location. Devices such as FIFOs are normally placed within such regions in the memory map to ensure that data is only read out when requested by the programmer, and therefore cannot be lost due to speculative fetches.

The mechanism that the Cortex-R4 processor uses to correct bit errors that occur in instruction fetches from Tightly Coupled Memory (TCM) involves re-reading and then (if necessary) writing the corrected data. Since 32-bit Thumb2 instructions can be present in two words in memory, then two word locations must be corrected. Performing such operations is safe for instructions, since these live in normal-type memory.

If the instruction with the error is the very last word in a TCM, then due to this erratum, the second correction will spill over into the next memory region, regardless of the type of that memory. The data read from this location, and any error associated with it (except TCM bit errors) are ignored, and no write is performed to this location, but this location could potentially be a read-sensitive device such as a FIFO, and therefore a spurious read could cause data corruption.

Conditions

1) The processor is configured with at least one TCM interface with error correction (does not apply to parity detection), and error checking and error correction (ECC) is enabled, and
2) An instruction is fetched from the last word in the TCM region (for example, if the TCM is 8KB in size, and located at 0x00008000, then the word at 0x00009ffc), and
3) The data fetched from the TCM contains a correctable (1-bit) error, and
4) The next memory location (the word at 0x0000a000 in this example) is in an MPU region that is marked as device or strongly-ordered type memory, and the data at that location is read-sensitive (for example, a FIFO).

Implications

This erratum only affects one or two words in the entire memory map (depending on the number and location of TCM interfaces in use). If it occurs, the effect of the erratum is to generate a spurious read from a read-sensitive device at one of these locations, which is likely to result in data corruption.

Workaround

A simple way of avoiding the erratum would be to ensure that there is no read-sensitive memory at either of the affected locations.

452032: Processor can deadlock when debug mode enables cleared

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The Cortex-R4 processor supports two different debugging modes: Halt-mode and Monitor-mode, or both modes can be disabled. Bits [15:14] in the Debug Status and Control Register (DBGDSCR) control which, if any, mode is enabled. Additionally, debug events can only occur if the invasive debug enable pin, DBGEN is asserted.

Debug events can be generated by the breakpoint or watchpoint units, matching instructions executed or data accesses performed. If there are active breakpoints or watchpoints at the time when the debugging modes are disabled via the DBGDSCR or DBGEN, this erratum can cause the processor to deadlock (in the case of a breakpoint) or lose data (in the case of a watchpoint).

Changing the debug mode enables while there are active breakpoints or watchpoints is considered UNPREDICTABLE. When a breakpoint is active and the debug mode enables are changed a deadlock may occur as a result of this erratum. This is considered erroneous because it is outside the bounds of permitted UNPREDICTABLE behavior.

Note: when a watchpoint is active and the debug mode enables are changed data loss may occur. This is considered acceptable because the scenario is UNPREDICTABLE.

The ARMv7 debug architecture permits the invasive debug enable, DBGEN, to be changed dynamically, that is, it may change at any time including while the processor is running. Deassertion of DBGEN can, because of this erratum, cause either a deadlock or data loss and both behaviors are considered erroneous because this scenario is architecturally permitted (not UNPREDICTABLE).

Conditions

1) DBGEN is asserted and the processor is running, and
2) At least one breakpoint or watchpoint is programmed and active, and
3) Either halt-mode debugging or monitor mode debugging is enabled, and
4) Either an instruction is fetched which matches a breakpoint, or an item of data is accessed which matches a watchpoint, and
5) After the instruction or data is accessed, but before the instruction completes execution, either the DBGEN input is deasserted or both halt-mode and monitor-mode debugging are disabled by means of a write the DBGDSCR.

Implications

Depending on which of the conditions are met, the processor will either lose data or deadlock. If the processor deadlocks because of this erratum it will still respond to interrupts provided they are not masked.

Workaround

This erratum can be avoided by ensuring that all watchpoints and breakpoints are made inactive before either deasserting DBGEN or changing the debug mode enables.

453266: Spurious error event may be generated on an instruction cache invalidate

Category C

Products Affected: Cortex-R4.

Present in: r1p1

Description

The Cortex-R4 processor contains a Performance Monitoring Unit that can be configured to count various events that happen, and an event bus interface to export these events to other components on the SoC.

Event number 0x4A counts the number of errors detected in the instruction cache tag RAMs.

The processor also contains a Correctable Fault Location Register (CFLR) that holds the location of the last correctable error detected.

Under the following conditions, the processor may incorrectly assert the event and incorrectly update the CFLR when no real error was present.

Conditions

1) The instruction cache is implemented and supports parity or ECC.
2) Error checking is enabled in the Auxiliary control register bits[5:3].
3) An invalidate instruction cache line by MVA (MCR p15, 0, Rn, c7, c5, 1) instruction is executed.
4) The address specified by the invalidate instruction is present in the instruction cache.
5) The data returned by the tag RAMs during the write part of the invalidate is such that it contains an error pattern that would have caused a parity or ECC error to be detected if the access had been a read.
6) An interrupt, imprecise abort, or debug request occurs whilst the invalidation instruction is being executed.

Condition 5 means that the behavior of the instruction-cache tag RAMs used has an effect on the erratum.

- If the RAMs are transparent when being written to, so that the data on the output is the same as the data being written, then they will never show an error pattern when being written to, hence this erratum cannot occur.
- If during a write the RAM outputs retain the value from the previous address that was read, then this erratum can only occur if the previous read detected an error. Therefore the CFLR may get updated twice for the same location, and the event may be generated twice, but it will have been caused by a real error.
- If during a write the value of the RAM outputs cannot be predicted, then the event may be triggered and the CFLR updated spuriously when there was no real error.

Implications

The implications of this erratum depend on the behavior of the instruction cache tag RAMs used in the specific implementation of the processor.

A real error may be reported twice, or an error that didn't exist may be reported. This erratum does not cause any extra prefetch or data aborts to be taken.

Workaround

There is no workaround available.

454166: TCM error events may be incorrectly signaled

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1

Description

The Cortex-R4 processor contains a Performance Monitoring Unit that can be configured to count various events that happen, and an event bus interface to export these events to other components on the SoC.

Event numbers 0x64, 0x65, 0x66, 0x67, 0x68, and 0x69 count the number of single and multi-bit errors detected on reads from each of the TCM ports. These event signals are valid 3 cycles after the last data cycle of a TCM transaction. These events may sometimes be signaled on speculative accesses.

In certain situations these events may signal an error spuriously.

In certain situations the relevant event bus signals may be signaled at the wrong time.

Conditions

There are a number of failure conditions. For any of the failures to be observed on the event bus the event bus must be enabled in the performance monitor control register. For any of the failures to be observed in the performance monitoring unit, one of the event counting registers must be configured to count the event.

The timing of the generated event may be wrong if the processor performs a read to a TCM interface, caused either by a load instruction, a store instruction that requires a read-modify-write sequence, or an AXI transaction received by the AXI-slave interface, and a subsequent data read is stalled in the pipeline or AXI-slave.

A spurious event may be generated if:

1) The processor performs a read to a TCM interface, caused either by a load instruction, a store instruction, or an instruction fetch, and the internal error checking is disabled for the relevant TCM interface in the second auxiliary control register.
2) The processor performs a read to a TCM interface, caused either by a load instruction or a store instruction, and the instruction either:
   - generates an MPU fault, or
   - fails its condition code, or
   - hits a watchpoint.
3) The processor performs two reads to a TCM interface, caused either by a load instruction or a store instruction, and the data returned by the first read contains an error.

Implications

In most systems, events generated by the processor are either counted for monitoring purposes, or cause particular behavior, such as the enabling of trace from an ETM. If an event is simply counted, then any error in the timing of that event caused by this erratum will have no noticeable effect. If system uses the event for any other purpose, the error may be noticeable, but since the timing error is small, it will not normally have a significant effect.

If this erratum causes spurious events to be generated, the result will be that the count of the event will be too large. Since event counts are considered to be approximate, this will not normally have a significant effect.

Workaround

For most systems, the event imprecision due to the erratum may be tolerable. If it is not, then use other events. There are events that report 1-bit and multi-bit errors from the LSU, PFU and AXI slave rather than the individual TCM interfaces.

459904: PMCR IDCODE field is incorrect

Category C
Products Affected: Cortex-R4.
Present in: r1p0

Description

The IDCODE field, bits [23:16] of the Performance Monitor Control Register (PMNC) read 0x00. The correct value is 0x14.

Implications

Any code attempting to identify the processor by reading this identification field will not be able to.

Workaround

There are a number of alternative identification fields in other CP15 register that can be used to identify the processor.

461513: Memory Model Feature Register 3 value is incorrect

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2

Description

The value in the Cache maintain MVA field, bits [3:0] of the Memory Model Feature Register 3 (MMFR3) is 0x0, but should be 0x1.

Implications

Software which tries to identify which cache maintenance operations are available on Cortex-R4 will incorrectly determine that the "by MVA" and "invalidate all instruction cache" operations are unavailable. This will lead the software to use "by set/way" operations instead, which will take longer, but will not result in any erroneous behavior.

Workaround

There is no workaround for this erratum.

463964: Access to TCM with wait cycles via AXI slave interface may livelock.

Category C

Products Affected: Cortex-R4.

Present in: r1p0

Description

The *TCWAIT signals on the Cortex-R4 TCM ports can be asserted, in the data phase of an access, to stall that access and cancel any following access that is in its address phase. The TCM controller might be built to assert the wait signal for a number of cycles for every access, so that every access takes a fixed number of cycles.

Cortex-R4 includes an AXI slave interface which allows other bus masters to access the Cortex-R4 TCMs. It is possible for the system to route an access from the Cortex-R4 AXI master interface back to the Cortex-R4 TCMs via the AXI slave interface. This erratum means that if either the LSU or PFU is performing two accesses to the TCM - one internally and the other via the AXI master and slave interfaces - then the two accesses can lock against each other if the TCM controller generates a fixed number of wait cycles.

Conditions

The LSU or PFU must perform data accesses/instruction fetches from two addresses as follows:

1) The first access must be to an address that is not marked as a TCM region, and therefore the request is made on the AXI master interface. Additionally, the system must route this access back to the Cortex-R4 AXI slave interface to access one of the TCMs attached to the Cortex-R4.

2) The second access must be a read to an address that is within the TCM address region for the same TCM port as the first fetch is accessing (ie. a "direct" access), and

3) The TCM controller must generate one or more wait cycles for every TCM access presented to it.

If the fetches are from the PFU, the two accesses must be performed as the result of sequential instruction execution. If the second fetch is the target of a taken branch executed from the first fetch (predicted correctly or not) then the livelock will not occur. This implies that the second fetch must be at the lowest address in the TCM region, and the first must be from the address immediately below that.

The requirement for the two accesses to the TCM to be direct, and via the AXI system implies that for this erratum to occur, the TCM must be mapped to different addresses internally and externally. That is, the TCM address space for the processor, as defined by the TCM Region Register, must differ from the address for the AXI-master, as defined by the bus decoders in the external system. In any system where these address spaces are the same, the erratum cannot occur.

Implications

If the erratum occurs, the processor will livelock. No instructions will be executed, and no AXI-transactions completed on the AXI-slave interface until an interrupt or debug entry request from the debugger occurs.

Workaround

This erratum can be avoided by ensuring that the processor never accesses the same TCM directly and via the AXI system at the same time. To prevent the program accessing the TCM via the AXI system, an MPU region can be programmed to prohibit access to the addresses at which the TCM appears in the external system. Any code which attempts to access the TCM via the AXI system will then generate an abort rather than livelock due to this erratum.

500166: On exit from Dormant Mode, data marked as dirty can be lost

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2

Description

The Cortex-R4(F) processor can be implemented in such a way as to support Dormant Mode. Dormant Mode is a power saving mode in which the processor logic, but not the processor TCM and cache RAMs, is powered down. The processor state, apart from the cache and TCM state, is stored to memory before entry into Dormant Mode, and restored after exit.

This erratum can cause the data cache to lose dirty data after a Dormant Mode cycle.

Conditions

1) The processor is configured with a data cache, and the cache is enabled.
2) A store has been performed to a memory location which is in a memory region marked as cacheable, write-back. Therefore at least one line in the data cache is dirty, that is, it contains data which has not been written back to the level 2 memory system.
3) The processor has been put into Dormant Mode, and returned to Run Mode.
4) After returning to Run Mode, but before any address in a write-back region is allocated to the data cache, an allocation to the data cache of an address in a write-through region causes a re-allocation of a dirty cache line.

Implications

If the Cortex-R4 processor is implemented to support Dormant Mode and all the conditions are met, a line of dirty data stored in the data cache is not written back to the level 2 memory system on eviction. This data is lost, which is likely to corrupt the program state. However, Dormant Mode was not an official feature of the processor before version r1p3, in which this erratum is fixed.

Workaround

If Dormant Mode is being used, the erratum can be avoided by using write-through caching only. This can be forced by setting bit [9] of the auxiliary control register.

Note: it is also possible to avoid this issue by ensuring that only write-back caching is used. However, there is no simple way of forcing this behavior.

Alternatively, if write-back caching is being used with Dormant Mode, after exit from Dormant Mode, and before enabling the data cache:

1) Perform a clean-and-invalidate data cache operation to an address in a write-back region (Note: if the MPU is disabled, then addresses in the range 0x0-0x3fffffff are suitable).
2) Enable the data cache, then perform a DMB.
3) Perform a load from the same address that was invalidated in (1), followed by a DMB.
4) Disable the data cache and perform another DMB.

After performing these operations, the processor state, including the cache enable, can be restored as required.

599517: CP15 Auxiliary ID and Prefetch Instruction accesses are UNDEFINED

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The ARMv7-R architecture requires implementation in CP15 of the following two features:

1) An Auxiliary ID Register (AIDR), which can be read in privileged modes, and the contents and format of which are IMPLEMENTATION DEFINED.

2) The operation to prefetch an instruction by MVA, as defined in the ARMv6 architecture, to be executed as a NOP.

Because of this erratum, both of these CP15 accesses generate an UNDEFINED exception on Cortex-R4.

Conditions

Either of the following instructions is executed in a privileged mode:

- MRC p15,1,<Rt>,c0,c0,7 ; Read IMPLEMENTATION DEFINED Auxiliary ID Register
- MCR p15,0,<Rt>,c7,c13,1 ; NOP, was Prefetch instruction by MVA in ARMv6

Implications

If software attempts to read the AIDR as part of its process of identifying the processor on which it is running, it will encounter an unexpected UNDEFINED exception.

If software attempts to execute an instruction prefetch using the ARMv6 CP15 prefetch operation, it will encounter an unexpected UNDEFINED exception.

Workaround

In the first case, because the contents of the AIDR are IMPLEMENTATION DEFINED, it must always be read in conjunction with the Main ID Register (MIDR). A simple way of avoiding this erratum would be to read and analyze the MIDR first and, if this indicates that the processor is Cortex-R4 skip the read of the AIDR.

In the second case, any instruction to prefetch an instruction by MVA should be removed or replaced by a true NOP instruction.

639819: An instruction which causes a data watchpoint to match is incorrectly traced

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

When tracing program execution using the ETM an extra instruction is traced if a data watchpoint matches and causes a debug exception. The extra instruction that is traced is the instruction which caused the data watchpoint to match.

Conditions

The extra instruction is traced if:

- a hardware watchpoint matches AND
- DBGEN is asserted AND
- monitor debug-mode is enabled

Implications

Trace analysis tools will incorrectly consider the instruction which causes a data watchpoint to match to have executed.

If any of the ETM address comparators are configured to match on address of the instruction and the exact match bit is set, the comparator will incorrectly fire. This might cause an unexpected trigger or change in any ETM resources which are configured to be sensitive to the address comparator.

Workaround

This is a workaround for trace tool vendors.

If a data abort exception is taken and the cause was a data watchpoint, the instruction traced immediately before the entry to the exception handler was not executed and must be discarded.

685018: Processor exits debug halt state when DBGEN is low

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The invasive debug enable input, DBGEN must be high in order for a debug event to put the processor into debug halt state. While in debug halt state, DBGEN may be deasserted, and this should have no effect. Because of this erratum, when DBGEN is deasserted, the Cortex-R4 processor will leave debug halt state as if DBGRESTART had been asserted and deasserted.

Conditions

- The Cortex-R4 processor is in debug halt state.
- The invasive debug enable input, DBGEN is deasserted.

Implications

With the Cortex-R4 processor, it is permitted to control the DBGEN signal dynamically, meaning that it might change while the processor is running, or while the processor is in Debug state. In particular, the DBGEN signal may be driven by a peripheral connected to the Cortex-R4. If a debugger connected to the Cortex-R4 has to change the DBGEN signal, it may need to first put the Cortex-R4 into debug halt state, then program the peripheral to deassert the DBGEN signal, and then finally restart the processor by exiting debug halt state.

As a consequence of this erratum, the processor will leave debug halt state early, when DBGEN is deasserted. Once this has happened, the debugger is not able to execute the correct sequence to restart the Cortex-R4, for example restoring any registers it has changed in order to program the peripheral driving DBGEN. This may lead to incorrect software behavior.

If DBGEN is statically driven, or can be deasserted without requiring the Cortex-R4 to be in debug halt state, then this erratum will not occur.

Workaround

The signal connected to the DBGEN should be connected to a chain of three registers, with the output of the third register connected to DBGEN in place of the original signal. The first two registers in this chain must be free running, while the third must be enabled only when DBGTRIGGER and DBGACK are both low. This logic ensures that when the processor is in debug halt state, or is about to enter debug halt state, the input DBGEN cannot change, and therefore the erratum is avoided.

720270: Latched DTR-full flags not updated correctly on DTR access

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

When the debug Data Transfer Register (DTR) is in non-blocking mode, the latched DTR-full flags (RXfull_l and TXfull_l) record the state of the DTR registers as observed by the debugger and control the flow of data to and from the debugger to prevent race hazards. For example, when the target reads data from DBGDTRRXint, the associated flag RXfull is cleared to indicate that the register has been drained, but the latched value RXfull_l remains set. Subsequent debugger writes to DBGDTRRXext are ignored because RXfull_l is set. RXfull_l is updated from RXfull when the debugger reads DBGDSCRext such that a debugger write to DBGDTRRXext will only succeed after the debugger has observed that the register is empty.

The ARMv7 debug architecture requires that RXfull_l is updated when the debugger reads DBGDSCRext and when it writes DBGDTRRXext. Similarly, TXfull_l must be updated when the debugger reads DBGDSCRext and when it reads DBGDTRTXext. In Cortex-R4, because of this erratum, RXfull_l and TXfull_l are only updated when the debugger reads DBGDSCRext.

Conditions

The DTR is in non-blocking mode, that is, DBGDSCR.ExtDCCmode is set to 0b00 and either,

1) The debugger reads DBGDSCRext which shows that RXfull is zero, that is, DBGDTRRX is empty and then
2) The debugger writes data to DBGDTRRXext, and
3) Without first reading the DBGDSCRext, and before the processor has read from DBGDTRRXint, the debugger performs another write to DBGDTRRXext.

or

1) The debugger reads DBGDSCRext which shows that TXfull is one, that is, DBGDTRTX is full and then
2) The debugger reads data from DBGDTRTXext, and then
3) The processor writes new data into DBGDTRTXint, and
4) Without first reading the DBGDSCRext, the debugger performs another read from DBGDTRTXext.

Implications

The ARMv7 debug architecture requires the debugger to read the DBGDSCRext before attempting to transfer data via the DTR when in non-blocking mode. This erratum only has implications for debuggers that violate this requirement. If the erratum occurs, data transfer and therefore data loss may occur, whereas the architecture requires that data transfer never occurs.

Workaround

There is no workaround for this erratum.

722412: CPACR.ASEDIS and CPACR.D32DIS incorrect when configured with floating-point

Category C

Products Affected: Cortex-R4F.

Present in: r1p0, r1p1, r1p2, r1p3

Description

In implementations of the VFPv3-D16 architecture that do not also implement the Advanced SIMD functionality, the ASEDIS and D32DIS bits, which are bits [31] and [30] respectively of the Coprocessor Access Control Register (CPACR), are read-as-one/writes ignored (RAO/WI), indicating that Advanced SIMD functionality and registers D16-D31 are disabled. Because of this erratum, these bits read zero in implementations of Cortex-R4F which include the floating-point unit.

When the floating point unit is not included, all bits of the CPACR correctly read-as-zero (RAZ).

Conditions

In an implementation of Cortex-R4F with the floating-point unit included, the CPACR is read.

Implications

If software uses the CPACR to determine whether Advanced SIMD functionality and registers D16-D32 are available, it will incorrectly determine that they are. Any attempt to use these features will lead to an unexpected UNDEFINED exception.

Workaround

Since the Cortex-R4F processor never includes Advanced SIMD functionality or registers D16-D31, this can be correctly determined by reading the part number from one of the ID registers rather than examining ASEDIS and D32DIS in the CPACR.

726554: DBGDSCR.ADAdiscard is wrong when DBGDSCR.DBGack set

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

When the DBGDSCR.ADAdiscard bit is set, asynchronous data aborts are discarded except for setting the DBGDSCR.ADAbort sticky flag. The Cortex-R4 processor ensures that all possible outstanding asynchronous data aborts have been recognized before it enters debug halt state, and therefore sets this flag immediately on entry to debug halt state to indicate that the debugger does not need to take any further action to determine whether all possible outstanding asynchronous aborts have been recognized.

Because of this erratum, the Cortex-R4 processor also sets the DBGDSCR.ADAdiscard bit when the DBGDSCR.DBGack bit is set. This can cause the DBGDSCR.ADAbort bit to become set when the processor is not in debug halt state, and it is not cleared when the processor enters debug halt state. However, the processor does not discard the abort, it is pended or generates an exception as normal.

Conditions

1) The processor is not in debug halt state.

2) The DBGDSCR.DBGack bit is set.

3) An asynchronous data abort (for example, SLVERR response to a store to Normal-type memory) is recognized.

Note: it is not expected that DBGDSCR.DBGack will be set in any Cortex-R4 system.

Implications

If this erratum occurs, and the processor subsequently enters debug halt state, the DBGDSCR.ADAbort bit will be set when in fact no asynchronous data abort has occurred in debug state. Before exiting debug state, the debugger will check this bit and will typically treat it as an error. If no other asynchronous data abort has occurred in debug state, then this is a false error.

Workaround

There is no workaround for this erratum.

732313: DBGPRSR sticky reset status may not be set on CPU reset

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The sticky reset status bit in the debug power and reset status register (DBGPRSR) records whether the processor has been reset since the last time the register was read. Because of this erratum, in some circumstances, the bit may not be set when the processor is reset.

Conditions

1) The DBGPRSR sticky reset status bit is clear AND,
2) The CPU reset, **nRESET** is asserted and then deasserted, AND either:
    1) There are no **PCLKDBG** clock edges during the time in which the CPU reset is asserted, OR
    2) The debug-APB clock enable signal, **PCLKENDBG** is not asserted during the time in which the CPU reset is asserted.

The debug-APB clock enable signal, **PCLKENDBG** is used to indicate which debug interface clock edges correspond to connected APB bus clock edges when, for example, the debug interface is clocked at twice the frequency of the connected bus. However, the Cortex-R4 debug interface has its own dedicated clock which in most systems will be connected to the bus clock with the clock enable tied HIGH. Condition 2-2 cannot occur in such a system.

Implications

If this erratum occurs then the debugger will not observe a CPU reset.

Workaround

There is no workaround for this erratum.

733066: CFLR reports correctable fault location inconsistently when multiple correctable faults occur simultaneously

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The Correctable Fault Location Register (CFLR) records the location of correctable ECC errors occurring in either the cache or Tightly Coupled Memories (TCMs). It records the set and way (for the cache) or the address and which TCM (for TCM) as well as whether it was an instruction fetch, data access or DMA (AXI-slave) access which generated the error. This information is intended to be polled when the event bus (EVNTBUS) signals a correctable error event.

In the unlikely event that two correctable errors are detected at the same time, only one of the errors is recorded in the CFLR using a fixed priority scheme. Because of this erratum, the information recorded may be inconsistent in this instance. Specifically, if simultaneous errors are detected by a data access and a DMA access, then the CFLR.Type field will record a d-side error but the rest of the CFLR will record the location of the DMA error.

Conditions

1) Cortex-R4 is built with the AXI-slave interface, at least one TCM port and ECC protection on at least one TCM, and
2) Cortex-R4 is built with a d-cache with either parity or ECC protection, and
3) ECC checking for the TCM is enabled (bits [27:25] of the auxiliary control register), and
4) D-cache error checking is enabled (bits [5:3] of the auxiliary control register are not 0b100), and
5) A data-side lookup detects a correctable error in the d-cache, at the same time as
6) An AXI-slave access detects a correctable error in one of the TCMs.

Implications

If logging software is recording correctable errors according to the source of the access (derived from the CFLR), then this erratum may cause small errors in the statistics. Note that, because the CFLR only records one error at a time, the logging software will never be able to construct a completely accurate picture of the numbers of errors occurring. The errors will typically be small because the occurrence of simultaneous random errors is rare.

If the logging software is only recording errors according to the RAM that the errors are detected in then the results will be correct despite this erratum.

Workaround

There is no workaround for this erratum.

736960: Debug halt exceptions are always shown as cancelling on ETM interface

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

When tracing program execution using the ETM, the instruction executed immediately before an external debug halt request will not be traced. If this instruction is a partially-executed multi-cycle instruction, for example a load-multiple which has transferred some, but not all of its registers, then this is correct. However, when the instruction has completed execution, this is erroneous.

Conditions

The erratum occurs if:

1) Tracing is enabled, and

2) The processor enters debug halt state either because EDBGRQ was asserted or because of a write to the DRCR, and

3) At the time it stopped executing instructions in order to enter debug halt state, the processor was not part-way through executing a load or store multiple instruction.

Implications

Trace analysis tools will incorrectly consider the instruction executed immediately before the debug halt state entry to have not executed.

If any of the ETM address comparators are configured to match on address of the instruction and the exact match bit is set, the comparator will fail to fire. This might have a knock-on effect to an expected trigger event or change in any ETM resources which are configured to be sensitive to the address comparator.

Workaround

There is no workaround for this erratum.

745322: DBGDSCR.SPNIDdis and DBGDSCR.SPIDdis are always zero

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

In the debug status and control register (DBGDSCR) the SPNIDdis and SPIDdis bits indicate whether non-invasive debug and invasive debug respectively are enabled. They reflect the values of the DBGENm and NIDENm inputs. Because of this erratum, these bits always read as zero on Cortex-R4, indicating that all debug is enabled, regardless of the value on the inputs.

Conditions

This erratum occurs when either:

1) The DBGDSCR.SPNIDdis bit (bit [17]) is read on CPUm when the DBGENm and NIDENm are both LOW, or

2) The DBGDSCR.SPIDdis bit (bit [16]) is read on CPUm when the DBGENm is LOW.

Note:

The SPNIDdis and SPIDdis bits are intended to be used in systems that implement the ARM security extensions. Cortex-R4 does not implement the ARM security extensions and therefore these bits are not expected to be used.

Implications

An external debugger or debug software running on the processor may determine that debug is enabled when it is actually disabled.

Workaround

An external debugger can determine whether invasive or non-invasive debug is enabled by reading bit [4] or bit [6] of the Authentication Status Register (DBGAUTHSTATUS).

748619: Missing reset exception on ETM interface

Category C
Products Affected: Cortex-R4.
Present in: r1p0, r1p1, r1p2, r1p3

Description

When tracing program execution through soft reset using the ETM a reset exception might not be traced.

Two sets of conditions cause this erratum to occur.

Conditions set 1

The first instruction at the reset vector is a load-store multiple instruction. This should not happen because after reset there is no base register whose contents can be guaranteed.

Conditions set 2

The processor enters debug halt state before executing the instruction at the reset vector due to the following conditions being true when the processor comes out of soft reset with nCPUHALTm HIGH or when nCPUHALTm is first de-asserted after soft reset

1) The Halting mode debug enable bit (bit[14] in the DBGDSCR) is set AND
2) The DBGENm input pin is asserted AND
3) A debug event is triggered due to:
   - The Halt request bit in the DBGDRCR being set OR
   - The EDBGRQm input pin being asserted

Implications

For conditions set 1, if the last instruction before the reset was an indirect branch instruction then a branch to the reset vector will be traced but not marked with a reset exception. If the last instruction before the reset was not an indirect branch then a trace analysis tool might incorrectly infer the execution of one or more instructions after the instruction before the reset until the processor executes an indirect branch. Typically the instruction at the reset vector is an indirect branch and therefore this error is limited to one or two instructions. The address comparators in the ETM are unaffected unless an address comparison was set on the address of the instruction just before the reset occurs and the exact match bit was set for comparison. In this case the instruction is always considered to be executed.

For conditions set 2, the reset exception is not traced and only the debug exception is traced. Any ETM address comparator configured to match on the instruction at the reset vector will not match.

Workaround

There is no work-around for this erratum.

754269: Register corruption during a load-multiple instruction at an exception vector

Category C
Products Affected: Cortex-R4.
Present in: r1p0, r1p1, r1p2, r1p3

Description

In certain circumstances, a load multiple instruction can cause corruption of a general purpose register.

Conditions

All the following conditions are required for this erratum to occur:

1) A UDIV or SDIV instruction is executed with out-of-order completion of divides enabled
2) A multi-cycle instruction is partially executed before being interrupted by either an IRQ, FIQ or imprecise abort. In this case, a multi-cycle instruction can be any of the following:
   - LDM/STM that transfers 3 or more registers
   - LDM/STM that transfers 2 registers to an unaligned address without write back
   - LDM/STM that transfers 2 registers to an aligned address with write back
   - TBB/TBH
3) A load multiple instruction is executed as the first instruction of the exception handler
4) The load multiple instruction itself is interrupted either by an IRQ, FIQ, imprecise abort or external debug halt request

This erratum is very timing sensitive and requires the UDIV or SDIV to complete when the load multiple is in the Issue stage of the CPU pipeline. The register that is corrupted is not necessarily related to the load-multiple instruction and will depend on the state in the CPU store pipeline when the UDIV or SDIV completes.

Implications

For practical systems, it is not expected that an interruptible LDM will be executed as the first instruction of an exception handler, because the handler is usually required to save the registers of the interrupted context. Therefore, it is not expected that this erratum has any implications for practical systems.

If the situation of the erratum occurs it will result in the corruption of the register bank state and could cause a fatal failure if the corrupted register is subsequently read before being written.

Workaround

To workaround this erratum, set bit [7] of the Auxiliary Control Register to disable out-of-order completion for divide instructions. Code performance may be reduced depending on how often divide operations are used.

756022: Watchpointed access in a store-multiple is not masked

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

Cortex-R4 implements synchronous watchpoints. A synchronous watchpoint generated during a store multiple instruction must ensure that the watchpointed accesses does not perform writes on the bus to update memory. Due to this erratum this requirement is not met and the processor will incorrectly update memory for a watchpointed access.

Conditions

All the following conditions are required for this erratum to occur:

1) A store multiple instruction is executed with at least 2 registers to be stored
2) The store multiple instruction writes to memory defined as Strongly-Ordered or Device
3) A watchpoint hit is generated for any access in the store multiple apart from the first access
4) The watchpoint hit is generated for an access with address bits[4:0] != 0x0

In these cases the store multiple will continue to perform writes until either the end of the store multiple or the end of the current cache line.

Implications

Due to this erratum, the memory contents of the watchpointed address are updated before the debug event can be recognized. This means that a debugger:

1) Cannot always assume memory has not been updated when a watchpoint generates a debug event
2) Cannot prevent an access by setting a watchpoint on it

The ARM architecture recommends that watchpoints should not be set on individual device or strongly ordered addresses that can be accessed as part of a load or store multiple. Instead, it recommends the use of the address range masking functionality provided to set watchpoints on an entire region, ensuring that the watchpoint event will be seen on the first access of a load or store multiple to this region.

If this recommendation is followed, this erratum will not occur.

Workaround

There is no workaround for this erratum.

758119: Incorrect read data may be returned for a load to the D-cache

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

Cortex-R4 implements an optional data-cache that, if implemented, can be enabled and disabled under the control of privileged software. Due to this erratum, under certain limited and timing-sensitive conditions, the act of enabling the D-cache may result in a subsequent load instruction returning incorrect data.

Conditions

All of the following conditions are required to stimulate this erratum.

1) The Cortex-R4 is implemented with a data-cache AND
2) Whilst the cache is disabled, one or more stores are performed to a region of memory that is defined as being Normal AND
3) The cache is subsequently enabled AND
4) A Load Exclusive instruction is performed such that all bytes read by the load were written by a combination of the previous stores and the address is to a Normal, Non-shared and Cacheable region of memory

The exact conditions for this erratum are very timing sensitive and require the store buffer slot containing the store data to be attempting to drain onto the bus when the linefill for the load exclusive is initiated. For this to happen, the load-exclusive instruction must occur just after the cache is enabled as otherwise the store buffer slot is likely to have drained.

When this erratum occurs, the load exclusive instruction returns the correct data but corrupts some internal processor state which may result in a subsequent load performed to a non TCM region of memory returning incorrect data.

Implications

Due to this erratum a load instruction may return incorrect data causing unrecoverable failure.

This erratum is considered to be very unlikely to occur in practice as the execution of stores to the same address as a subsequent load exclusive is unlikely to occur around the D-cache being enabled.

Workaround

This erratum can be avoided by executing a DSB instruction either before or after the D-cache is enabled but before any subsequent accesses to Cacheable regions of memory.

Note that the pair of instructions to enable the D-cache and execute a DSB are not atomic and an interrupt can be taken between them. This means that the code executed in the interrupt handlers must also be considered when evaluating if this erratum can occur. If required, interrupts must be masked when enabling the D-cache.

758269: Watchpoint on a load or store multiple may be missed

Category C

Products Affected: Cortex-R4.

Present in: r1p0, r1p1, r1p2, r1p3

Description

Cortex-R4 supports synchronous watchpoints. This implies that for load and store multiples, a watchpoint on any memory access will generate a debug event on the instruction itself. Due to this erratum, certain watchpoint hits on multiples will not generate a debug event.

Conditions

All the following conditions are required for this erratum to be stimulated

1) A load or store multiple instruction is executed with at least 5 registers in the register list AND
2) The address range accessed corresponds to Strongly-Ordered or Device memory AND
3) A watchpoint match is generated for an access that does not correspond to either the first two or the last two registers in the list

Under these conditions the processor will lose the watchpoint. Note that for a store multiple instruction, the conditions are also affected by pipeline state making them timing sensitive.

Implications

Due to this erratum a debugger may not be able to correctly watch accesses made to Device or Strongly-ordered memory.

The ARM architecture recommends that watchpoints should not be set on individual Device or Strongly ordered addresses that can be accessed as part of a load or store multiple. Instead, it recommends the use of the address range masking functionality provided to set watchpoints on an entire region, ensuring that the watchpoint event will be seen on the first access of a load or store multiple to this region.

If this recommendation is followed, this erratum will not occur.

Workaround

There is no work around for this erratum.