ARM Processor
Cortex^{TM}-R5 and Cortex-R5F

Product Revision r1

Software Developers Errata Notice

Non-Confidential - Released

Software Developers Errata Notice

Web Address

http://www.arm.com

Feedback on content

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

- the document title
- the document number, ARM-EPM-012129
- 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.

08 Jan 2016: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
14	New	853474	CatB	Rare	Self-modifying code in non-cacheable memory might not work with a slow memory system

20 Jun 2012: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
12	New	780125	CatB	Rare	Processor might deadlock or lose data when configured with cache-ECC

24 Apr 2012: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
8	New	752270	CatB		A store exclusive performed to the AXI peripheral port may return an incorrect status
9	New	761669	CatB		ACP coherency can fail when relying only on address dependency ordering or out-of-memory communication ordering
11	New	772721	CatB	Rare	In Standby Mode, peripheral port may lose read transactions
15	New	753369	CatC		Register corruption during a load-multiple instruction at an exception vector
16	New	756523	CatC		Watchpointed access in a store-multiple is not masked
17	New	758471	CatC		Watchpoint on a load or store multiple may be missed
18	New	758472	CatC		Incorrect read data may be returned for a load to the D-cache
19	New	773269	CatC		Correctable TCM ECC errors can additionally signal fatal TCM-centric events

Contents

Chapter 1. **Introduction**

This chapter introduces the errata notice for the ARM Cortex-R5 and Cortex-R5F 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
761669	CatB		ACP coherency can fail when relying only on address dependency ordering or out-of-memory communication ordering	X	X		
752270	CatB		A store exclusive performed to the AXI peripheral port might return an incorrect status	X			
853474	CatB	Rare	Self-modifying code in non-cacheable memory might not work with a slow memory system	X	X	X	X
780125	CatB	Rare	Processor might deadlock or lose data when configured with cache-ECC	X	X	X	X
772721	CatB	Rare	In Standby Mode, peripheral port may lose read transactions	X	X	X	
773269	CatC		Correctable TCM ECC errors can additionally signal fatal TCM-centric events	X	X	X	
758472	CatC		Incorrect read data might be returned for a load to the D-cache	X	X		
758471	CatC		Watchpoint on a load or store multiple might be missed	X	X		
756523	CatC		Watchpointed access in a store multiple is not masked	X	X	X	X
753369	CatC		Register corruption during a load-multiple instruction at an exception vector	X			

2.3. Category A

There are no errata in this category

2.4. Category A (Rare)

There are no errata in this category

2.5. Category B

752270: A store exclusive performed to the AXI peripheral port might return an incorrect status

Category B

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0

Description

Cortex-R5 can perform exclusive accesses on its AXI peripheral interface and virtual AXI peripheral interface in response to an exclusive access instruction. A store exclusive instruction should return a status field that depends on the response of the exclusive access on the bus. Because of this erratum, the status field returned might be incorrect.

Configurations Affected

Systems with memory attached to the Cortex-R5 AXI peripheral port which is shared by other masters. This includes systems with one memory accessed by both CPUs in a twin-CPU Cortex-R5 processor.

Conditions

1) The relevant AXI peripheral interface must be enabled and
2) A STREX or STREXB or STREXH instruction must be executed with the following properties:
   - The destination address must be in a shared memory region as defined by the MPU and
   - The destination address must be within the relevant AXI peripheral interface region of the memory map and
   - The internal exclusive access monitor must be in the exclusive state when the store exclusive instruction is executed.

Implications

Exclusive accesses to shared memory locations are typically used in semaphores to coordinate multiple masters accessing the same memory location. Because of this erratum, the store exclusive part of the exclusive pair always appears to have failed. This is likely to cause livelock, because software is typically written to wait indefinitely until the store exclusive passes.

Note that exclusive accesses to non-shared memory locations are unaffected by this erratum. Therefore, single master systems that only need synchronization between multiple threads or processes running on the same CPU can function correctly.

Workaround

There are two workarounds for this erratum:

1) Use non-shared memory for all exclusive accesses on the AXI peripheral and virtual AXI peripheral interfaces. This is possible if the memory in question does not need to be accessible by multiple masters in the system.
2) Use SWP and SWPB instead of the exclusive instructions to implement the required synchronization.

761669: ACP coherency can fail when relying only on address dependency ordering or out-of-memory communication ordering

Category B

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1

Description

Cortex-R5 implements an accelerator coherency port (ACP) which enables coherency between processor non-cached or write-through cached regions and read/writes performed through said ACP port.

The coherency scheme is designed to allow a master attached to the ACP to determine that any write it performs is visible to the processor once the ACP port returns the BRESP acknowledgment associated to said write.

This permits an ACP attached master to signal the processor, via an interrupt or a subsequent memory write, that it can now safely and coherently access any data previously written by said master, or to guarantee the order of visibility of memory writes to the processor.

An erratum exists within the Cortex-R5 store-buffer which can, under timing and transaction dependent conditions, result in the processor being temporarily able to read out-of-date data previously written by the core but subsequently over-written by a transaction through the ACP, producing a condition incompatible with the previously described coherency scheme.

The processor architecture requires the use of a data memory barrier (DMB) to enforce load/store ordering in the majority of cases and the execution of a DMB rectifies the stale-data issue within the store-buffer. As a result, the erroneous effects of this erratum are therefore limited to two specific cases:

1) An interrupt issued to, and taken by, the processor in response to an ACP write BRESP acknowledgement is not sufficient by itself to guarantee that all ACP writes before said BRESP will be visible to code executing in the processor's interrupt handler.

2) A pair of ordered ACP writes, requiring the address request of one to have been performed after receiving the BRESP acknowledgement of the other, may expose the defect if subsequently accessed via a pair of loads on the processor whose order is guaranteed only by an address dependency. That is to say the data from the first load matches the data from the second ACP write, the address of the second load is a function of the data from the first load and the value read by the second load is from the same memory location as that written by the first ACP write; under this scenario, the erratum makes it possible for the processor to potentially return out-of-date data for the second load.

The effect of this erratum can be masked through draining the store-queue by the insertion of a single DMB (or DSB) instruction either, in case 1, at any point between entering the interrupt handler and reading a memory location written by the ACP, or, in case 2, between the first and second load.

Conditions

The erratum requires one of the following coherent interactions between the processor (P1) and an external ACP connected master (E) to be used in observing memory ordering between P1 and E:

1. Interrupt enforced ordering:

```
P1:
  Write A
E:
  Read A (see P1's)
  Write A
  DSB (see BRESP for write)
  IRQ P1
P1:
  Take IRQ
  Read A (can see P1's)  // Should always see E's
```

or:

2. Address dependency ordering:

```
P1:
  Write A


E:
  Read A (see P1's)
  Write A
  DMB (see BRESP for write)
  Write B


P1:
  Read B (see E's)
  <address dependency between Read B and Read A>
  Read A (can see P1's)  // Should always see E's
```

Note that the conditions additionally require a rare timing interaction resulting from P1 both reading and writing within the same 64-bit granule as A in close temporal proximity to E's "Write A", i.e. the sequences above executed in isolation cannot trigger the failure.

Implications

Under the conditions listed above, it is not possible to rely on the visible ordering between ACP and processor accesses being coherent when enforced only via an interrupt triggered in response to an ACP BRESP, or via address-dependency ordered loads from locations written in order via the ACP.

Workaround

For interrupt related cases, the addition of a DMB between entering the interrupt handler and accessing any data written by the ACP will re-establish order.

For address-dependency ordered processor loads from locations written via the ACP, the addition of a DMB between the loads will re-establish order.

2.6. Category B (Rare)

772721: In Standby Mode, peripheral port may lose read transactions

Category B Rare

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1, r1p2

Description

The Cortex-R5 processor implements 32-bit AXI and AHB peripheral ports which can be used as an alternative master for all types of memory transactions apart from instruction fetches and certain doubleword accesses. The processor also implements Standby Mode, entered by executing a Wait-For-Interrupt (`WFI`) or Wait-For-Event (`WFE`) instruction. In Standby Mode the processor stops executing instructions and also stops the clock to most of the logic. Because of this erratum, if an AXI peripheral port read transaction has been interrupted it can be continually presented on the bus or it can ignore responses from the slave.

Configurations Affected

The processor has Normal memory attached to the AXI peripheral port (as distinct from Devices).

Conditions

1) **DBGNOCLKSTOP** is not asserted, and
2) The CPU initiates a read via the AXI peripheral interface or the AXI virtual peripheral interface at an address which is marked as Normal-type memory, and
3) Before the read has completed, the CPU recognizes and takes an asynchronous exception (interrupt, asynchronous abort or debug entry request), and
4) The CPU subsequently executes `WFI` or `WFE` to enter Standby Mode, and
5) Either:
   1. The address for the transaction has not been accepted before the CPU gates its clock or,
   2. The read data for the transaction is returned while the CPU has its clock gated or,
   3. The CPU is reset and the read data is returned after the CPU has restarted.

Note: Because the CPU discards any read data, and because the AHB protocol requires the slave to retain **HREADY** asserted when the bus is idle, this erratum does not affect the AHB peripheral interface.

Implications

If this erratum occurs, data read from Normal memory on the AXI peripheral port might be corrupted or the CPU might deadlock.

This erratum cannot occur in systems where interrupt handlers and asynchronous abort handlers return to re-execute the interrupted instruction (that is, they do not switch context), and the handler code does not itself execute `WFI` or `WFE`.

Workaround

It is possible to avoid this erratum by marking the whole of the AXI peripheral interface address space as Device-type memory. If the CPU is configured with an MPU, you can achieve this by programming the MPU region registers appropriately.

Alternatively, you can work around this erratum by executing a load to a peripheral port address upon entering any interrupt or asynchronous abort handler routine, before any `WFI`/`WFE` or the exception return.

780125: Processor might deadlock or lose data when configured with cache-ECC

Category B Rare

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1, r1p2, r1p3

Description

The Cortex-R5 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.

853474: Self-modifying code in non-cacheable memory might not work with a slow memory system

Category B Rare

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1, r1p2, r1p3

Description

If a program executing on a processor writes to a non-cacheable memory location that it subsequently executes an instruction from, it must issue DSB and ISB to ensure that the updated instruction is executed. Because of this erratum this sequence might not be sufficient to prevent the processor executing the old instruction or a corrupted instruction.

Configurations Affected

This erratum affects all configurations. However, it can only occur if the memory system connected to the AXI-master port is capable of returning some data for a read burst and then completing a write to the same location before the last beat of the read burst is returned.

Conditions

In order for this erratum to occur, the following steps must occur, in the order 1-2-3-4 or 2-1-3-4, but not necessarily consecutively, i.e. other instructions may be executed between the steps listed:

1) The processor initiates a speculative instruction fetch from a 32-byte aligned block which is in non-cacheable memory (that is, the MPU has marked it as non-cacheable or the instruction cache is disabled) that is not in TCM.

2) The processor executes a store which writes to a location within the block (but not within the last n-bytes of the block, where n-bytes is the width of the memory servicing the request). The write occurs in the memory after the read to the same location has been returned.

3) The processor executes DSB-ISB as required by the architecture.

4) The processor executes an instruction which was written by the store, before the initial read transaction completes. This implies that there is a noticeable delay between completion of the individual beats of the transaction in [1].

Note: The processor can only fetch instructions on the AXI-master port from one 32-byte aligned block at a time. If any instruction is executed between [3] and [4] which is in a different 32-byte aligned block to [1] and is non-cacheable or misses in the instruction cache, then this erratum will not occur. This implies that the modified instruction must be executed soon after it has been modified.

Implications

When this erratum occurs the processor will execute a stale instruction or a corrupt instruction, which is likely to result in incorrect operation of the program, potentially including program flow corruption.

Workaround

This erratum can be avoided by ensuring that at least one instruction, fetched from non-cacheable memory which is not in the same 32-byte naturally aligned block as the modified instruction, is executed between the ISB which synchronizes the code modification and the modified instruction. ARM recommends inserting a BLX to such a location which contains a BX lr that will return immediately.

2.7. Category C

753369: Register corruption during a load-multiple instruction at an exception vector

Category C
Products Affected: Cortex-R5, Cortex-R5F.
Present in: r1p0

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 three or more registers
   - LDM/STM that transfers two registers to an unaligned address without write back
   - LDM/STM that transfers two registers to an aligned address with write back
   - TBB/TBH
3) A load multiple instruction executes 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 depends on the state in the CPU store pipeline when the UDIV or SDIV completes.

Implications

For practical systems, it is unlikely that an interruptible LDM executes 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 unlikely that this erratum has any implications for practical systems.

If this erratum occurs it will corrupt 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 might be reduced depending on how often divide operations are used.

756523: Watchpointed access in a store multiple is not masked

Category C

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1, r1p2, r1p3

Description

Cortex-R5 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. Because of this erratum this requirement is not met and the processor incorrectly updates memory for a watchpointed access.

Conditions

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

1) A store multiple instruction executes with at least two 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 except for the first access
4) The watchpoint hit is generated for an access with address bits[4:0] != 0x0

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

Implications

Because of 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 not to set watchpoints 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 is seen on the first access of a load or store multiple to this region.

This erratum will not occur if you follow this recommendation.

Workaround

There is no workaround for this erratum.

758471: Watchpoint on a load or store multiple might be missed

Category C

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1

Description

Cortex-R5 supports synchronous watchpoints. This implies that for load and store multiples, a watchpoint on any memory access generates a debug event on the instruction itself. Because of this erratum, certain watchpoint hits on multiples do not generate a debug event.

Conditions

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

1) A load or store multiple instruction is executed with at least five 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 loses the watchpoint. Note that for a store multiple instruction, the conditions are also affected by pipeline state, making them timing sensitive.

Implications

Because of this erratum a debugger might 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 is seen on the first access of a load or store multiple to this region.

This erratum will not occur if you follow this recommendation.

Workaround

There is no work around for this erratum.

758472: Incorrect read data might be returned for a load to the D-cache

Category C

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1

Description

Cortex-R5 implements an optional data-cache that, if implemented, can be enabled and disabled under the control of privileged software. Because of this erratum, under certain limited and timing-sensitive conditions, the act of enabling the D-cache might result in a subsequent load instruction returning incorrect data.

Conditions

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

1) The Cortex-R5 is implemented with a data-cache and
2) While 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 because 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 might 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.

773269: Correctable TCM ECC errors can additionally signal fatal TCM-centric events

Category C

Products Affected: Cortex-R5, Cortex-R5F.

Present in: r1p0, r1p1, r1p2

Description

Each CPU in the Cortex-R5 processor generates a number of events which are visible on the primary outputs **EVNTBUSm** and can be counted by the performance monitor event counters. A number of these events indicate when ECC errors have been detected by the CPU, different events distinguishing whether the errors occurred in ATCM, B0TCM, B1TCM or cache, and whether they are correctable or fatal. TCM errors generate two sets of events, CPU-centric events which indicate the actual errors the CPU dealt with, and TCM-centric events which have a fixed timing relative to the TCM access but may be signaled on some speculative accesses.

Because of this erratum the CPU might, in rare circumstances, signal a TCM-centric fatal error event spuriously. This erratum does not affect CPU-centric error events and TCM-centric correctable error events.

Conditions

The following conditions are all required for this erratum to occur:

1) The Cortex-R5 processor is configured with at least one TCM port with 64-bit ECC, and ECC checking and correction is enabled.
2) Out-of-order completion of integer divide, single-precision floating point divide or square root or double-precision floating point arithmetic instructions is enabled. This is the default setting.
3) The AXI-slave interface is being used to access a TCM which is configured with 64-bit ECC.

The remaining conditions relate to the micro-architectural state of the CPU and are therefore complex to determine at the system or programmer's level. These conditions render the erratum very rare and include:

1) The erratum can only occur when a 1-bit ECC error is detected on an instruction fetch from the TCM.
2) The erratum can only occur when a read from the AXI-slave interface has been denied access to the TCM for fifteen access cycles because the CPU i-side or d-side have made back-to-back accesses to the same TCM and these take priority.

If these conditions occur, the CPU incorrectly signals a fatal ECC error event using **EVNTBUSm[39:37]** and by incrementing the performance monitor if the corresponding event (0x64, 0x65 or 0x66) is selected. This event is in addition to the correctable ECC error events generated from the detected error which are correctly signaled for the instruction fetch and the associated correction.

Implications

Many systems use error events for logging or statistics generation. In these systems this erratum means that the generated statistics are slightly inaccurate. This might lead, for example, to self-test routines being run more frequently than would otherwise be expected.

Systems which use fatal error events to initiate more invasive behaviours, such as restarting or shutting down the CPU, are expected to use the CPU-centric events which do not signal on speculative accesses. Such systems will not be affected by this erratum.

Workaround

This erratum can be avoided by disabling all out-of-order instruction execution, which can be controlled using the CP15 Auxiliary Control Register. Set ACTLR.sMOV (bit [10]) and, if the CPU is configured with floating-point, ACTLR2.DOODPFP and ACTLR2.DOOFMACS (bits [17:16]). If you use this workaround it might reduce the performance of code which uses floating point or divide instructions.