



ARM Core  
ARM11 MPCore (MP002)  
**Errata Notice**

This document contains all errata known at the date of issue in releases up to and including revision r2p1 of MPCore

**Proprietary notice**

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM Limited in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

**Document confidentiality status**

This document is Non Confidential.

**Web address**

<http://www.arm.com/>

**Feedback on the product**

If you have any comments or suggestions about this product, contact your supplier giving:

- The product name
- A concise explanation of your comments.

**Feedback on this document**

If you have any comments on about this document, please send email to <mailto:errata@arm.com> giving:

- The document title
- The documents number
- The page number(s) to which your comments refer
- A concise explanation of your comments

General suggestion for additions and improvements are also welcome.

## Contents

INTRODUCTION	6
ERRATA SUMMARY TABLE	11
ERRATA - CATEGORY 1	13
364261: An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system	13
452050: Incorrect hazard checking on L1 cache evictions can cause out-of-date data read	15
720247: Speculative Instruction fetches can be made anywhere in the memory map	17
ERRATA - CATEGORY 2	19
342696: Priority mask register of the interrupt controller interface is reset to the wrong value	19
342698: CPU exiting from STANDBY mode can deadlock	20
351420: Externally aborted PLDs can cause silent failure	22
351422: Spinlocks using LDREX and STREX instructions can livelock	24
351704: The VFP11 can exhibit a data corruption in cases involving a double-precision multiply or multiply-accumulate operation	27
396433: VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction	29
403347: Jazelle Security	31
408881: Inappropriate hazard checking on virtual address can cause read-after-write hazard	32
408883: non 64-bit aligned VFP load-store multiple can cause data corruption or UNDEFINED exception trap	33
415588: A CPU coherent linefill can incorrectly be flagged as non-coherent to the SCU	34
725514: Registers related to TLB lockdown entries are always read as zero	36
ERRATA - CATEGORY 3	38
336141: VFP can take an inexact exception on non-CDP VFP instructions	38
340351: WFE during a debug request is executed twice	39
364373: Data corruption is possible if AXI memory system is not maintaining transaction order.	41
446466: S bit value in TTBRn is not reflected on ARUSER[0] signal on page table walks	43
498365: ETM11 prevents CPU clock stopping invoked by WFE for a CPU that it is connected to in an MPCore system	45
501614: SCU access can be incorrectly denied to all CPUs through the SCU Control Register	46

---

<b>717874:</b>	<b>No Vector Catch debug event on reset when High Vectors are enabled</b>	<b>47</b>
<b>721879:</b>	<b>TLBIMVAA operation may not invalidate all required TLB entries when multiple page sizes are used.</b>	<b>48</b>
<b>ERRATA - DOCUMENTATION</b>		<b>49</b>
<b>341802:</b>	<b>TRM must state that BTAC is not automatically flushed when FCSE PID or Context ID registers are changed</b>	<b>49</b>
<b>ERRATA – DRIVER SOFTWARE</b>		<b>51</b>
<b>There are no Errata in this Category</b>		<b>51</b>
<b>ERRATA – IMPLEMENTATION</b>		<b>52</b>
<b>447678:</b>	<b>SCU duplicated TAG RAM BIST data is presented in incorrect order</b>	<b>52</b>

## Introduction

### Scope

This document describes errata categorised 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

### Categorisation of Errata

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

Category 1      Behavior that is impossible to work around and that severely restricts the use of the product in all, or the majority of applications, rendering the device unusable.

Category 2      Behavior that contravenes the specified behavior and that might limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Category 3      Behavior that was not the originally intended behavior but should not cause any problems in applications.

## Change Control

### 22 Dec 2009: Changes in Document v25

Page	Status	ID	Cat	Summary
17	Updated	720247	Cat 1	Speculative Instruction fetches can be made anywhere in the memory map
36	New	725514	Cat 2	Registers related to TLB lockdown entries are always read as zero
47	Updated	717874	Cat 3	No Vector Catch debug event on reset when High Vectors are enabled
48	Updated	721879	Cat 3	TLBIMVAA operation may not invalidate all required TLB entries when multiple page sizes are used.

### 04 Sep 2009: Changes in Document v24

Page	Status	ID	Cat	Summary
17	New	720247	Cat 1	Speculative Instruction fetches can be made anywhere in the memory map
47	New	717874	Cat 3	No Vector Catch debug event on reset when High Vectors are enabled
48	New	721879	Cat 3	TLBIMVAA operation may not invalidate all required TLB entries when multiple page sizes are used.

### 17 Jul 2008: Changes in Document v23

Page	Status	ID	Cat	Summary
43	Updated	446466	Cat 3	S bit value in TTBRn is not reflected on ARUSER[0] signal on page table walks
45	Updated	498365	Cat 3	ETM11 prevents CPU clock stopping invoked by WFE for a CPU that it is connected to in an MPCore system
46	Updated	501614	Cat 3	SCU access can be incorrectly denied to all CPUs through the SCU Control Register
52	Updated	447678	Impl	SCU duplicated TAG RAM BIST data is presented in incorrect order

### 11 Mar 2008: Changes in Document v22

Page	Status	ID	Cat	Summary
15	New	452050	Cat 1	Incorrect hazard checking on L1 cache evictions can cause out-of-date data read
43	New	446466	Cat 3	S bit value in TTBRn is not reflected on ARUSER[0] signal on page table walks
45	New	498365	Cat 3	ETM11 prevents CPU clock stopping invoked by WFE for a CPU that it is connected to in an MPCore system
46	New	501614	Cat 3	SCU access can be incorrectly denied to all CPUs through the SCU Control Register
52	New	447678	Impl	SCU duplicated TAG RAM BIST data is presented in incorrect order

### 12 Dec 2006: Changes in Document v20

Page	Status	ID	Cat	Summary
29	Updated	396433	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction

31	Updated	403347	Cat 2	Jazelle Security
32	Updated	408881	Cat 2	Inappropriate hazard checking on virtual address can cause read-after-write hazard
34	New	415588	Cat 2	A CPU coherent linefill can incorrectly be flagged as non-coherent to the SCU
33	Updated	408883	Cat 2	non 64-bit aligned VFP load-store multiple can cause data corruption or UNDEFINED exception trap
38	Updated	336141	Cat 3	VFP can take an inexact exception on non-CDP VFP instructions

**20 Nov 2006: Changes in Document v16**

Page	Status	ID	Cat	Summary
31	New	403347	Cat 2	Jazelle Security
32	New	408881	Cat 2	Inappropriate hazard checking on virtual address can cause read-after-write hazard
33	New	408883	Cat 2	non 64-bit aligned VFP load-store multiple can cause data corruption or UNDEFINED exception trap

**11 Jul 2006: Changes in Document v15**

Page	Status	ID	Cat	Summary
29	Updated	396433	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction

**07 Jul 2006: Changes in Document v14**

Page	Status	ID	Cat	Summary
29	New	396433	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction

**10 Feb 2006: Changes in Document v13**

Page	Status	ID	Cat	Summary
13	Updated	364261	Cat 1	An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system
41	Updated	364373	Cat 3	Data corruption is possible if AXI memory system is not maintaining transaction order.

**12 Jan 2006: Fixed Date Of Issue field v12****10 Nov 2005: Changes in Document v11**

Page	Status	ID	Cat	Summary
13	New	364261	Cat 1	An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system
41	New	364373	Cat 3	Data corruption is possible if AXI memory system is not maintaining transaction order.



**10 Nov 2005: Changes in Document v10**

Page	Status	ID	Cat	Summary
13	New	364261	Cat 1	An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system
41	New	364373	Cat 3	Data corruption is possible if AXI memory system is not maintaining transaction order.

**09 Nov 2005: Changes in Document v9**

Page	Status	ID	Cat	Summary
13	New	364261	Cat 1	An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system
41	New	364373	Cat 3	Data corruption is possible if AXI memory system is not maintaining transaction order.

**19 Sep 2005: Changes in Document v8**

Page	Status	ID	Cat	Summary
22	Updated	351420	Cat 2	Externally aborted PLDs can cause silent failure
24	Updated	351422	Cat 2	Spinlocks using LDREX and STREX instructions can livelock
27	Updated	351704	Cat 2	The VFP11 can exhibit a data corruption in cases involving a double-precision multiply or multiply-accumulate operation
49	Updated	341802	Doc	TRM must state that BTAC is not automatically flushed when FCSE PID or Context ID registers are changed

**05 Jul 2005: Changes in Document v7**

Page	Status	ID	Cat	Summary
22	New	351420	Cat 2	Externally aborted PLDs can cause silent failure
24	New	351422	Cat 2	Spinlocks using LDREX and STREX instructions can livelock
27	New	351704	Cat 2	The VFP11 can exhibit a data corruption in cases involving a double-precision multiply or multiply-accumulate operation
49	Updated	341802	Doc	TRM must state that BTAC is not automatically flushed when FCSE PID or Context ID registers are changed

**10 Mar 2005: Changes in Document v4**

Page	Status	ID	Cat	Summary
19	New	342696	Cat 2	Priority mask register of the interrupt controller interface is reset to the wrong value
20	New	342698	Cat 2	CPU exiting from STANDBY mode can deadlock
39	New	340351	Cat 3	WFE during a debug request is executed twice
49	New	341802	Doc	TRM must state that BTAC is not automatically flushed when FCSE PID or Context ID registers are changed
38	Updated	336141	Cat 3	VFP can take an inexact exception on non-CDP VFP instructions

---

**13 Dec 2004: Change control Correction v3.0****03 Dec 2004: Corrected Date of Issue field v2.0****03 Dec 2004: Document Creation v1.0**

Page	Status	ID	Cat	Summary
------	--------	----	-----	---------

38	New	336141	Cat 3	VFP can take an inexact exception on non-CDP VFP instructions
----	-----	--------	-------	---

## Errata Summary Table

The errata associated with this product affect product versions as below.

A cell shown thus **X** indicates that the defect affects the revision shown at the top of that column.

ID	Cat	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r1p0	r2p0	r2p1
341802	Doc	TRM must state that BTAC is not automatically flushed when FCSE PID or Context ID registers are changed		X					
364261	Cat 1	An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system	X	X	X				
452050	Cat 1	Incorrect hazard checking on L1 cache evictions can cause out-of-date data read	X	X	X				
720247	Cat 1	Speculative Instruction fetches can be made anywhere in the memory map	X	X	X	X	X	X	
342696	Cat 2	Priority mask register of the interrupt controller interface is reset to the wrong value	X						
342698	Cat 2	CPU exiting from STANDBY mode can deadlock	X						
351420	Cat 2	Externally aborted PLDs can cause silent failure		X					
351422	Cat 2	Spinlocks using LDREX and STREX instructions can livelock	X	X					
351704	Cat 2	The VFP11 can exhibit a data corruption in cases involving a double-precision multiply or multiply-accumulate operation	X	X					
396433	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction	X	X	X	X			
403347	Cat 2	Jazelle Security	X	X	X	X			
408881	Cat 2	Inappropriate hazard checking on virtual address can cause read-after-write hazard	X	X	X	X			
408883	Cat 2	non 64-bit aligned VFP load-store multiple can cause data corruption or UNDEFINED exception trap	X	X	X	X			
415588	Cat 2	A CPU coherent linefill can incorrectly be flagged as non-coherent to the SCU	X	X	X	X			
725514	Cat 2	Registers related to TLB lockdown entries are always read as zero	X	X	X	X	X	X	

ID	Cat	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r1p0	r2p0	r2p1
336141	Cat 3	VFP can take an inexact exception on non-CDP VFP instructions	X	X	X	X	X	X	X
340351	Cat 3	WFE during a debug request is executed twice	X						
364373	Cat 3	Data corruption is possible if AXI memory system is not maintaining transaction order.	X	X	X				
446466	Cat 3	S bit value in TTBRn is not reflected on ARUSER[0] signal on page table walks	X	X	X	X	X		
498365	Cat 3	ETM11 prevents CPU clock stopping invoked by WFE for a CPU that it is connected to in an MPCore system					X		
501614	Cat 3	SCU access can be incorrectly denied to all CPUs through the SCU Control Register					X		
717874	Cat 3	No Vector Catch debug event on reset when High Vectors are enabled	X	X	X	X	X	X	
721879	Cat 3	TLBIMVAA operation may not invalidate all required TLB entries when multiple page sizes are used.	X	X	X	X	X	X	
447678	Impl	SCU duplicated TAG RAM BIST data is presented in incorrect order	X	X	X	X	X		

## Errata - Category 1

### **364261: An MP11 CPU can deadlock after issuing many outstanding buffered write transactions on high latency memory system**

#### **Status**

Affects: product MPCore.

Fault status: Cat 1, Present in: r0p0,r0p1,r0p2, Fixed in r0p3.

#### **Description**

Each MP11 CPU in an MPCore core is able to handle eight outstanding write transactions to normal non-cacheable memory regions.

#### **Conditions**

1. An MP11 CPU has its MMU turned on and set so that a memory region is defined as Normal Non-Cacheable region, as Normal write-through region (treated as non-cacheable in MPCore), or as Shared Normal write-back write-allocate with CPU in AMP mode.
2. The program being executed causes multiple write transactions to that memory region.
3. Eight write transactions have been issued and accepted by the memory system. In the case of MPCore, a new transaction is requested only when the last data quantity of the previous transaction has been accepted. Note also that eight transactions are not necessarily equivalent to the execution of eight store instructions as write accesses can be merged in the MP11 store buffer.
4. A ninth write transaction is issued by the MP11 CPU before the buffered write response of the first outstanding write has been received back from the memory system. MP11 CPUs are able to produce one buffered write every two CPU cycles, so in the worst case, the maximum latency of the buffered response on the bus is 16 CPU cycles, or 8 bus cycles if bus traffic is slowed down by ACLKEN usage.

#### **System design consideration**

The occurrence of this erratum depends on the system design as the erratum does not appear if the system cannot handle more than eight active write transactions. An active write transaction is one for which the address has been accepted, some or all of the data have been accepted, but the buffered write response has not been received yet.

Note that adding some AXI register slice between the MPCore AXI master ports and the accessed slaves can increase the number of supported active write transactions.

#### **Implications**

Once a ninth write transaction is requested by an MP11 CPU, that CPU is not able to complete any more write transactions, even after reception of the buffered response for the first eight outstanding requests, and it then eventually deadlocks.

**Workaround**

The software workaround for that erratum is to prevent write access to normal non-cacheable memory by changing the memory region attributes to Strongly ordered memory. That has a significant impact on the system performance.

**452050: Incorrect hazard checking on L1 cache evictions can cause out-of-date data read****Status**

Affects: product MPCore.

Fault status: Cat 1, Present in: r0p0,r0p1,r0p2, Fixed in r0p3.

**Description**

Incorrect hazard checking in SCU for L1 cache evictions can cause incorrect transaction ordering on AXI bus, causing out-of-date data read.

**Conditions**

1. An MP11 CPU has a dirty cache line in its L1 data cache.
2. This line sits in a memory region described as write-back write allocate shared, ie is considered as coherent.
3. The CPU that contains the cache line, and at least a second CPU in ARM11 MPCore core are in SMP mode, ie part of the coherency cluster.
4. This dirty line is being evicted due to normal line replacement or by a cache maintenance operation. The eviction is not completed yet because the associated write transaction has not yet received its AXI buffered response, or because the SCU master port it should go through is currently handling a previous transaction.
5. That same line is requested by another CPU taking part in the coherence.
6. As the line eviction has started, the SCU considers that the line is not in the first CPU cache, and so a linefill is requested to memory system through SCU master port.
7. Since the eviction write transaction is still waiting for its AXI buffered response, the SCU will incorrectly issue the linefill transaction on SCU master port. If no address hazard checking is performed in the memory sub-system, the linefill read transaction can overtake the eviction write transaction, causing out-of-date data fetching.
8. In the case where the eviction write transaction is stalled on a SCU master bus, ie the address has not yet been presented on the bus, the linefill read transaction can start on the second SCU master bus so that, even if address hazard checking is performed in the memory system, the linefill read transaction will overtake the eviction write transaction, causing out-of-date data fetching.

**Implications**

The incorrect address hazard checking on eviction can cause read data corruption.

**Workaround**

There is no software work-around for this erratum, however note that the erratum is not present in revisions later than r0p2.



**720247: Speculative Instruction fetches can be made anywhere in the memory map****Status**

Affects: product MPCore.

Fault status: Cat 1, Present in: r0p0,r0p1,r0p2,r0p3,r1p0,r2p0, Fixed in r2p1.

**Description**

Under some circumstances, ARM11 MPCore can request some speculative fetches to the Level 1 Instruction memory system that could ultimately cause an external read transaction to the underlying memory system. In rare cases, this speculative fetch is performed to a random address and cause problem if that address sits in a read-sensitive memory region where reads to a location are not idempotent.

**Conditions**

1. MMU is off, or is on but read-sensitive memory region don't have XN protection set in their respective page table descriptors
2. The system design permits instruction fetches to access devices which are read-sensitive.
3. Either:
  - The ARM11 MPCore is executing a instruction modifying the PC (Program counter) that is followed by:
    - a. A data processing or load operation with PC as destination. Note that this operation can be a real program instruction or the equivalent opcode of the operation (resulting in the concatenation of two Thumb Instructions, or in the value of data sitting in literal pool).
    - b. Internal events such as return Stack prediction evaluation or unresolved branch recovery

Or:

- The ARM11 MPCore is executing a load to a register followed within two instructions by a branch to that register, and interrupts or asynchronous external aborts are enabled and are being used

**Implications**

As the speculative instruction fetch is made to random address, there is a rare possibility that a Instruction transaction read request will be made to a read-sensitive device, leading to corruption of that read-sensitive behaviour.

**Workaround**

There is no robust software work-around for this erratum when the MMU is off, but depending on the memory system, a hardware workaround in the system could be applied by protecting all read-sensitive devices in the memory system so that they are responding by an error to all Instruction read request. In a system using AXI, this can be achieved by aborting accesses to the device if ARPROT[2] of the memory request indicates an instruction access

When the MMU is turned on, if the VMSAv6 page table format is being used with sub-pages disabled in the SCTRL, the problem is worked around by marking all read-sensitive memory regions as XN.

There is no workaround when the MMU is on if sub-pages are enabled.

## Errata - Category 2

### **342696: Priority mask register of the interrupt controller interface is reset to the wrong value**

#### **Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0, Fixed in r0p1.

#### **Description**

In the Distributed Interrupt Controller, the CPU interface Priority Mask register reset value is 0x00000000, but should be 0x000000F0 as described in MPCore TRM.

#### **Implications**

A reset value of 0x00000000 for the Priority Mask register in CPU interface means that, by default, all interrupts are masked by the CPU interface.

This erratum is unlikely to cause any problem as robust software would set this mask before enabling the distributed interrupt controller.

#### **Workaround**

No software work-around is necessary for this erratum.

**342698: CPU exiting from STANDBY mode can deadlock****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0, Fixed in r0p1.

**Description**

Under extremely rare conditions, a CPU exiting STANDBY mode can deadlock because the WFE or WFI instruction used to enter STANDBY mode stopped the CPU clock at inappropriate time.

**Conditions**

1. A memory access (Access 0) in the LSU is TLB aborted.
2. That memory access is followed by one or two other memory accesses in the LSU (Access1 and Access2 respectively).
3. At least one of Access1 or Access2 misses in the main TLB, starting a hardware page table walk. The accessed region by Access1 or Access2 must be defined as a page so that a second-level descriptor read is needed.
4. Access 0 abort makes the CPU execute its abort handler.
5. If the abort handler is executed fast enough (resides in I-cache) and the memory system is slow enough, the CPU will return from abort exception before the completion of the first-level page descriptor access caused by Access1 and/or Access2 main TLB miss.
6. A WFE/WFI instruction enters Ex1 pipeline stage before the first-level descriptor access is completed (knowing that a second-level descriptor access is pending).
7. The CPU stops its clock at the end of first-level descriptor access as it sees the L1 system ready to enter STANDBY state (BIU is idle for two cycles between end of first-level descriptor access and beginning of second-level descriptor access).
8. Before the clocks are stopped, the BIU has time to request the second-level descriptor access so that when the second-level descriptor data comes back from the SCU, the CPU don't see it as its clocks are stopped.
9. The system then deadlocks when the CPU exits STANDBY state as the CPU has a pending page table walk that will never complete.

**Implications**

This erratum is not expected to be seen by many applications as WFE and WFI instructions are only used in very particular and controlled pieces of software where MMU faults and external aborts are very unlikely to happen.

## Workaround

Software work-arounds exist for both WFE and WFI but are different as any work-around for WFE must be as light-weight as possible in order to keep the advantage of WFE.

The WFI software work-around is based on the principle that any pending hardware page table walk must be completed before a WFI instruction is executed.

This work-around consists of the following code sequence that has to be inserted in front of any WFI instruction:

```
MCR p15, 0, <Rd>, c7, c10, 4    ; Drain Synchronization Barrier
MCR p15, 0, <Raddr>, c8, c7, 3  ; Invalidate TLB Single Entry on MVA Only
PLD <Rd>, [<Raddr>]             ; Preload data at address contain by <Rn>
                                ; Entry previously invalidated in TLB
```

The Drain Synchronization Barrier guaranties that the aborts due to previous accesses are handled by the LSU and the CPU.

The Invalidate TLB Single Entry on MVA Only is here to guarantee that the following PLD instruction will cause a TLB miss and a subsequent hardware page table walk. No restriction exists for the address used for TLB entry invalidation.

PLD will TLB miss and so ensure that any previous hardware page table walk due to a killed slot is finished. Using a PLD instead of a LDR prevents restrictions on address used, as if PLD is aborted, it is not reported.

For WFE, the work-around consists of setting an un-documented bit in Auxiliary Control Register in order to override WFE instruction. Writing to bit 29 of Auxiliary Control Register has the effect of forcing MP11 CPU Event output signal to 1, meaning that a WFE instruction will always see that the CPU event register is set, and that the CPU will never enter STANDBY mode.

**351420: Externally aborted PLDs can cause silent failure****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p1, Fixed in r0p2.

**Description**

The ARM architecture state that PLD instruction never generates a data abort, nor does it signal any sort of memory exception detected for the accessed address in any other way. MPCore memory system has been designed so that PLD instructions are not blocking level-one memory system, so that data being fetched in a linefill buffer by a PLD instruction can be read or written to before the PLD instruction eventually completes.

That means that, with memory system for which the granularity of external aborts is finer than one line, a data can be read and written to before the PLD access is aborted. In case of data abort on a PLD, no exceptions are reported to the integer core, and the line sitting in the linefill buffer is discarded. So if that data has been read or written to, the integer core has no information that the read access was corrupted or that the write access has been discarded.

**Conditions**

For read accesses:

1. A PLD instruction execution causes a linefill that is aborted on any of the AXI read burst beat.
2. A LSU slot receives a read access from the integer core to a data lying in the line currently being linefilled by the PLD.
3. The LSU hits in the linefill buffer and gives the data back to the integer core without reporting abort information.

For write accesses:

1. A PLD instruction execution causes a linefill that is aborted on any of the AXI read burst beat.
2. A slot of the write buffer contains a write access to the line currently being linefilled by the PLD.
3. The write buffer merges the data of the slot in the linefill buffer.

**Implications**

Erratum implication for a write access is that, once the data has been merged in the linefill buffer and the PLD was externally aborted, the "dirty" linefill buffer is flushed (not allocated in the cache) and the write access is lost without any abort reported to the integer core.

For read accesses, the erratum implication is that some corrupted data are read back in the integer core without any abort reported.

**Workaround**

In the case the memory system attached to the ARM11 MPCore is able to generate error response on the bus (external aborts) and as PLD is a hint instruction, the only work-around for this erratum is to not use this instruction.

If no external aborts can be generated by the memory system, no preventive action is needed.

**351422: Spinlocks using LDREX and STREX instructions can livelock****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

**Description**

Under extremely rare conditions, in an MPCore node consisting of at least 3 CPUs, two CPUs trying to perform a STREX to data on the same shared (V6 memory attribute phrasing) cache line (including the case where they try to perform a STREX to the same data) can enter a livelock situation, if the code they're running is organised in the same way as the following.

```
void get_spinlock(sem_t *spinlock) {
    tmp_sem_t tmp;
    tmp_sem_t taken = TAKEN;
    tmp_sem_t free  = FREE;
    tmp_sem_t success = SUCCESS;
    __asm {
get_lock_loop:
        LDREX    tmp, [spinlock];
        CMP      tmp, free;
        BNE      get_lock_loop;
        STREX    tmp, taken, [spinlock];
        CMP      tmp, success;
        BNE      get_lock_loop;
    };
}
```

**Conditions**

1. A first CPU (CPU0) is performing a write (Which can consist of any of the derivatives from STR, STM, STC and STREX) to a data located on a cache line @A. This cache line is in the Shared state (MESI phrasing, meaning that it is present and clean in at least one cache among an SMP cluster), resulting into a coherency request (COH0) forcing the invalidation of this line in the other caches. COH0 is sent to the SCU, pending until it is serviced.
2. A second CPU (CPU1) is performing a STREX (or any size derivative of a STREX) to the same cache line @A (including the case where both CPU0 and CPU1 are writing to the same data), which is also shared (MESI phrasing) in its cache. This STREX results into a coherency request (COH1) forcing the invalidation of this line in the other caches. COH1 is sent to the SCU, pending until it is serviced.
3. The coherency request COH0 is arbitrated first, eventually resulting in the invalidation of the cache line @A in the cache of CPU1. As a result of this, the cache line @A is linefilled into the cache of processor CPU0.
4. A third CPU (CPU2) is performing a LDREX to the same cache line @A (including the case where CPU2 is performing a LDREX to the same data as CPU1 and/or CPU0). The cache line is linefilled into



its cached, the MESI state being SHARED (between at least CPU0 and CPU2). It then performs a STREX to this address, resulting into a coherency request (COH2) being issued to invalidate the cache line @A in the caches from the other CPUs. COH2 is pending at the SCU interface waiting to be serviced.

5. COH1 is then serviced by the SCU. Cache line @A having been invalidated in the mean time in the cache from processor CPU1, this results in CPU1 linefilling the cache line @A, and in the STREX enforcing a FAIL result in the result register. It also results in cache line @A being invalidated in CPU2's cache, which eventually will enforce the STREX from CPU2 to fail later on. CPU1 restarts the Software routine which consists first in a LDREX to the same data.
6. COH2 is then serviced by the SCU. Cache line @A having been invalidated in the mean time in the cache from processor CPU2, this results in CPU2 linefilling the cache line @A, and in the STREX enforcing a FAIL result in the result register. It also results in cache line being invalidated in CPU1's cache, which eventually will enforce the STREX from CPU1 to fail latter on. CPU2 restarts the Software routine which consists first in a LDREX to the same data.
7. CPU0 is spinning on a data on the same cache line @A. This forces a coherency request and a linefill for this line, which it gets in the MESI Shared state (shared with CPU1).
8. CPU1 performs the STREX to @A. This results in a coherency request (COH1') being issued to invalidate the cache line @A in the caches from the other CPUs.
9. CPU0 is spinning on a data on the same cache line @A. This forces a coherency request and a linefill for this line, which it gets in the MESI Shared state (shared with CPU1).
10. CPU2 performs the STREX to @A. This results in a coherency request (COH2') being issued to invalidate the cache line @A in the caches from the other CPUs.
11. The whole lot restarts from point 5) onward, according to the routine described in paragraph Description.
12. There are no external events that can disturb the oscillation described above (interruptions, memory accesses by another CPU on an irregular basis).

## Implications

This erratum is not expected to be seen by many applications as the situation described above requires a precise timing with just a window of a few clock cycles.

The CPUs must also be running with the interruptions disabled.

## Workaround

Three different software workaround exist:

- 1) The first workaround for a livelock is to ensure that an external event will disturb the system, often in the way of an interrupt arising on a regular basis (Clock tic of an OS).
- 2) In case interrupts are disabled or if the occurrence of an interrupt is not guarantied, the routine must ensure no oscillating case can occur. The following correction ensures such a livelock cannot occur.

```
void get_spinlock(sem_t *spinlock)
```

```
{
    tmp_sem_t tmp;
    register int j;
    register int i = get_cpuid();
    tmp_sem_t taken = TAKEN;
    tmp_sem_t free  = FREE;
    tmp_sem_t success = 0;
    __asm {
        B      get_lock_loop;
try_again:
        ADD    i, i, #1
        CMP    i, #5
        MOVEQ   i, #1
        MOV    j, i
wait_loop:
        NOP
        SUBS    j, j, #1
        BNE     wait_loop
get_lock_loop:
        LDREX   tmp, [spinlock];
        CMP     tmp, free;
        BNE     get_lock_loop;
        STREX   tmp, taken, [spinlock];
        CMP     tmp, success;
        BNE     try_again;
    };
}
```

3) The use of SWP instead of LDREX / STREX, in case the LDREX / STREX based routine is used to gain access to a spinlock, overcomes the livelock.

**351704: The VFP11 can exhibit a data corruption in cases involving a double-precision multiply or multiply-accumulate operation****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

**Description**

If a change of instruction stream occurs immediately after a floating point double-precision multiply or multiply-accumulate operation ("multiply"), then under very rare circumstances a following floating point operation which depends on the result of the multiply will get an incorrect value.

The erratum arises from a clearing of the dependency information in the VFP for the multiply as a result of a branch which is taken early in the ARM pipeline. The erratum causes a floating point operation at the target of a branch to be issued and executed instead of being stalled waiting on the result of the multiply. As a result, the input operand to this second floating pointing operation will not contain the result of the multiply.

A very similar form of the erratum occurs if the second floating point operation has the same destination register as the multiply, and the second floating point operation is one of FABS, FNEG or FCPY. In this case, the result of the second operation is written before the result of the multiply, so resulting in incorrect behaviour.

**Conditions**

The following two sets of conditions may give rise to this erratum

Condition Set 1:

1. Scalar mode operation
2. An FMSCD, FNMSCD, FMACD, FNMACD, FMULD or FNMULD instruction
3. One of the next two instructions is a mispredicted conditional branch that is folded
4. Branch prediction and branch folding are enabled
5. The predicted target of the branch must be a floating point operation which is not dependent on the multiply
6. The alternative target of the branch must have a floating point instruction within 3 instructions
7. The first floating point instruction after the alternative target of the branch must be dependent on the result of the multiply
8. None of the floating point instructions fail their condition code check

OR

Condition Set 2:

1. Scalar mode operation
2. An FMSCD, FNMSCD, FMACD, FNMACD, FMULD or FNMULD instruction
3. The next instruction is a MOV PC, Register, Shift #Imm4
4. The instruction immediately after the MOV PC, Register, Shift #Imm must be a floating point operation which is not dependent on the multiply

5. The target of the MOV PC must have a floating point instruction within 3 instructions
6. The first floating point instruction after the target of the MOV PC must be dependent on the result of the multiply
7. None of the floating point instructions fail their condition code check

It should be noted that particular timings of the arrival of instructions is required in addition to the above set of conditions for the erratum to be triggered. As a result, not all sequences that meet these criteria will trigger the erratum. In particular, the reliance on a mispredicted folded branch is likely to significantly reduce the likelihood of seeing this erratum repeatedly or consistently.

## Implications

The erratum only affects double-precision (i.e. not single precision) floating point multiplies. This condition can cause the destination register of the target instruction to be written with incorrect data, and no indication of the error will be seen. In systems where a very rare error in floating point operation can be tolerated, for example, a system using floating point for graphics, no workaround should be needed. In other systems, where the very occasional error is not acceptable, one of the workarounds must be employed.

The instruction MOV PC, Register, Shift #Imm is not one which is expected to be generated by a compiler, and is one that is of little use in any code sequence. It is not expected that this cause of the erratum will be seen in any real systems.

## Workaround

For Condition Set 1:

1. As a folded branch is required to trigger this form of the erratum, the Auxiliary Control Register bit BIT[3] should be used to disable branch folding. This is expected to result in only a small loss of performance.
2. For hand written VFP assembler code, it is possible to add any other two instructions (ie not a floating point multiply, branch or MOV PC with a shift) between the floating point multiply and the branch to work around this erratum.
3. For hand written VFP assembler code, it is possible to add a non-floating point instruction after the branch and at the destination of the branch. This will ensure that the branch is not folded onto a floating point instruction.

For Condition Set 2:

The MOV PC, Register, Shift #Imm instruction is very unlikely to be used in any real code sequences and is certainly very unlikely to be generated by a compiler. Therefore, the use of this instruction directly after double-precision multiplies should be avoided by assembler programmers in order to workaround this erratum. This can be achieved by inserting any other instruction (ie not a floating point multiply, branch or a MOV PC with a shift) between the floating point multiply and the MOV PC.

**396433: VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r1p0.

**Description**

The VFP reports exceptions on data processing operations imprecisely, that is, it can allow subsequent (non-data processing) instructions to execute while the exception status of an instruction is being computed. The scoreboard in the VFP guards against the completion of any subsequent instruction which overwrites any of the source operands of the previous data processing instruction. This ensures that when the exception is reported, the operands are available for the instruction generating the exception, and so it can be emulated by the exception handler.

In certain cases this hazard checking fails, and the VFP can allow an instruction to complete that overwrites the source operands of an exceptional operation. The result is that when the exception is taken the source operands can be corrupted and therefore the emulated result can be incorrect.

**Conditions**

1. The VFP is not configured to use the VFP11 RunFast mode. That is, one (or more) of the following conditions is not true:
  - FlushToZero is enabled;
  - DefaultNaN is enabled;
  - None of the exception enable bits in the FPSCR (FPSCR bits 15, 12-8) are set.
2. A VFP data processing instruction is generating an exception.
3. Either:
  - Vector mode is being used and a VFP Load or Move from ARM registers is within 3 instructions of the VFP data processing instruction OR Vector mode is not being used and a VFP Load or Move from ARM registers is within 2 instructions of the VFP data processing instruction.
4. The VFP Load or Move has a destination register which is one of the source registers of the VFP data processing instruction.

**Implications**

The following operations can be bounced to software:

1. When FlushToZero mode is not configured, data processing instructions where some element of the calculation results in a number (before rounding or underflow) which is not zero, but whose magnitude is less than:
  - $2^{-95}$  for single precision calculations OR
  - $2^{-959}$  for double precision calculations
2. Any data processing operation where an input is subnormal, that is smaller than:

- $2^{126}$  for single precision OR
- $2^{1022}$  for double precision calculations

The exact conditions are described in the VFP11 section of the ARM11 MPCore Processor TRM. In cases 1 and 2, the calculations are handled by software emulation, and because of this erratum, the result of the calculation can be incorrect.

Calculations will not exhibit this problem where all parts of the calculation only use zero or numbers whose magnitude is greater than:

- $2^{95}$  (approximately  $10^{29}$ ) for single precision OR
- $2^{959}$  (approximately  $10^{289}$ ) for double precision

In many applications that use VFP11, it is expected that the ranges of floating point numbers in use will meet this criteria and so this erratum is unlikely to be seen in practical situations.

When DefaultNaN mode is not configured, data processing operations, where an input of the calculation is a NaN (Not a Number), can be bounced to software. As a result of this erratum, the result of the calculation may be incorrect.

When the exceptions for Invalid Operation, Divide by Zero, Overflow, Input Subnormal or Underflow are enabled, data processing instructions which cause these exceptions can, as a result of this erratum, present the incorrect initial state to the exception handler. As many of these exceptions are typically used for debugging unexpected results, this results in incorrect information in determining the cause of the exception.

## Workaround

This erratum can be completely avoided if the VFP is run with:

1. FlushToZero enabled, and
2. Default NaN enabled, and
3. None of the exception enable bits in the FPSCR (FPSCR bits 15, 12-8) set.

However, this is not suitable for applications that require full IEEE754 compatibility, as it removes the capability to use subnormal numbers.

The erratum can also be avoided by ensuring the separation between the exceptional data processing instruction and the subsequent floating-point load or move that overwrites its source operands is great enough to avoid the problem. This might be achieved by re-arranging the code or by the insertion of NOPs into the code stream according to the following guidelines:

1. In scalar code, ensuring at least one instruction between a data processing instruction and any instruction which reloads any of the data processing instruction's source operands will alleviate the problem.
2. In vector code, ensuring at least two instructions between a data processing instruction and any instruction which reloads any of the data processing instruction's source operands will alleviate the problem.

**403347: Jazelle Security****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r1p0.

**Description**

An error in the Jazelle logic creates a vulnerability in any Java VM software which exploits the Jazelle acceleration hardware.

**Conditions**

ARM core is executing Java code in Java state.

**Implications**

The vulnerability allows a malicious program to read and write data to arbitrary memory locations within the VM memory space, which may cause the VM to crash or create further security vulnerabilities. Although it is difficult and unlikely, it may also be possible to exploit this vulnerability to execute arbitrary code. Exploitation of this vulnerability requires specific detailed information of both the security flaw and the VM implementation

- This errata only affects Java VM software which is designed to exploit the Jazelle hardware acceleration technology and is wholly contained within the Jazelle logic.
- Normal non-Jazelle operation is unaffected.

**Workaround**

A modification to the Java VM software provides a full and complete correction for the problem. An ARM patch is available for JTEK-K that contains the fixes for this vulnerability.

Details of the workaround are available to licensees of the JTEK software or the ARM Jazelle Architecture.

**408881: Inappropriate hazard checking on virtual address can cause read-after-write hazard****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r1p0.

**Description**

In the ARM11 MPCore CPU load-store unit, address comparisons are performed between slots in order to prevent read-after-write hazard. These comparisons are performed on the whole virtual address (bit [31:3]) instead of on the non-translated part of the address (bits [11:3]). In the case two virtual memory addresses are mapped to the same physical address, a read-after-write hazard may occur as the virtual addresses do not match.

**Conditions**

This problem occurs when the following conditions are met:

1. A write access is done while store buffer is full, meaning it stays in the load-store unit.
2. A read access is done on a different virtual address, mapping on the same virtual address as the store one.
3. No hazard is detected so the read is not hitting in the store buffer, so the read data is returned from the data cache or external memory.

**Implications**

The implication of that erratum is to return out-of-date data on a read access. Note that, in addition to conditions listed above, this erratum can only happen with two back-to-back memory transfers to different virtual address being aliased to the same physical address, a very unlikely software scenario.

**Workaround**

The software work-around is to prevent, if possible, any aliasing of virtual addresses to a single physical address. If this can't be avoided, extra care would be taken when accessing the two virtual addresses:

<LIST>

Use a memory barrier between the store and the load access.

Ensure that the two accesses are not performed back-to-back.

</LIST>



**408883: non 64-bit aligned VFP load-store multiple can cause data corruption or UNDEFINED exception trap****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r1p0.

**Description**

Non 64-bit aligned VFP load-store multiple can cause data corruption or UNDEFINED exception trap when immediately followed by a second VFP load-store multiple transaction.

**Conditions**

This problem occurs when the following conditions are met:

1. A load-store multiple VFP transaction (FLDM or FSTM) is done to a 32-bit aligned memory address.
2. Immediately following, another similar transaction is done, irrespective of 32 or 64-bit address alignment.

**Implications**

This erratum implication is a data corruption or an UNDEFINED trap.

**Workaround**

The only software work-around for this erratum is to ensure that all multiple accesses are always performed to 64-bit aligned addresses.

**415588: A CPU coherent linefill can incorrectly be flagged as non-coherent to the SCU****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r1p0.

**Description**

In a multi-CPU implementation, when at least two CPUs are in SMP mode, two consecutive linefills caused by a store buffer drain to a coherent memory region followed by a L1 data cache read miss to non-coherent region will cause the first one to be seen as non-coherent by the SCU.

**Conditions**

This problem occurs when the following conditions are met:

1. At least two CPUs of ARM11 MPCore are configured in SMP mode. This implies that single-CPU implementations of ARM11 MPCore are immune to this erratum.
2. One CPU is handling data in memory regions marked as write-back write-allocate shared and write-back write-allocate non-shared simultaneously.
3. That CPU performs a write access to a write-back write-allocate shared (coherent) region, and that write access misses in the CPU L1 data cache.
4. That CPU performs a consecutive write access to a write-back write-allocate non-shared (non-coherent) region, and that write access also misses in the CPU L1 data cache too.
5. That CPU performs a consecutive read access to a write-back write-allocate non-shared region that misses in the L1 data cache once more.
6. The linefill request to the CPU BIU for the above read access is requested exactly one cycle after the linefill request from the store buffer for the first write access to the write-back write-allocate shared region.

**Implications**

As the coherent linefill is not seen by the SCU, the SCU won't perform a lookup in its duplicated TAG RAMs to check if the accessed cache line is present or not in another CPU and will always request the data from the L2 memory sub-system.

If the accessed cache line is actually lying in another CPU L1 data cache in a modified state, the line being allocated in the CPU data cache will be corrupted.

The ARM SMP Linux port does not support the simultaneous use of both memory types required for this erratum and so is not expected to be affected by this erratum.

**Workaround**

There are two software workarounds for that erratum:

1. Ensure that software is not making mixed accesses to write-back write-allocate shared and non-shared regions at the same moment in time.
2. Declare all write-back write-allocate regions as shared. This could have a slight impact on linefill access latency due to the fact that any L1 data cache linefill request will cause a SCU duplicated TAG lookup before it is requested to the L2 memory sub-system.

**725514: Registers related to TLB lockdown entries are always read as zero****Status**

Affects: product MPCore.

Fault status: Cat 2, Present in: r0p0,r0p1,r0p2,r0p3,r1p0,r2p0, Fixed in r2p1.

**Description**

Main TLB Lockdown VA Main TLB Lockdown PA and Main TLB Lockdown Attribute registers are always read as zero, whichever Main TLB Lockdown entry is selected for read access.

**Conditions**

1. At least one Main TLB lockdown entry has been set up.
2. A valid Main TLB Lockdown entry is selected for read, and Main TLB Lockdown VA Main TLB Lockdown PA and Main TLB Lockdown Attribute registers are read using the following code sequence:

```
MCR p15, 5, <Rn>, c15, c4, 2 ; Select Lockdown TLB Entry for Read
MRC p15, 5, <Rm>, c15, c7, 2 ; Read Lockdown TLB attributes Register
MRC p15, 5, <Ri>, c15, c5, 2 ; Read Lockdown TLB VA Register
MRC p15, 5, <Rj>, c15, c6, 2 ; Read Lockdown TLB PA Register
```

**Implications**

The implication of this erratum is that if the software running on ARM11 MPCore has lost the knowledge of the values programmed in Main TLB Lockdown entries, it can't recover these values from the actual registers.

It must be noted that although the registers are always read as zero, the Main TLB lockdown entries contain correct programmes values and that the TLB hit and translation mechanisms are functional.

**Workaround**

TLB Lockdown operations have been designed in order to ease saving and restoring of lockdown entries when entering or exiting a CPU low-power state.

This erratum is impacting the saving part of this process as the lockdown entries content is not accessible. So a work-around would consist in keeping a copy of the values stored in the Main TLB Lockdown entries so that they can be recovered from external memory locations when CPU is exiting low-power state.



## Errata - Category 3

### **336141: VFP can take an inexact exception on non-CDP VFP instructions**

#### **Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r1p0,r2p0,r2p1, Open.

#### **Description**

If the VFP FPSCR register IXE bit is set, all VFP CDP instructions are bounced to software using an Undefined instruction trap. This also sets the VFP EXC register EX bit to cause most other following VFP operations to be exceptional. Because of this erratum, if a VFP CDP instruction appears in the shadow of a branch or exception, and the IXE bit is set, then the EX bit is set erroneously. This causes subsequent VFP instructions (other than CDP operations) to be bounced incorrectly.

#### **Conditions**

1. The VFP is enabled
2. The VFP is trapping inexact exceptions, by having the IXE bit set
3. A VFP CDP operation (or equivalent bit pattern) appears in a branch or exception shadow

#### **Implications**

This erratum is not expected to be seen by many applications, because the use of the VFP IXE bit to trap all inexact exceptions results in no VFP arithmetic operations being executed on the VFP. Instead, these operations are handled by the software support code. Consequently, very few, if any, applications, have the IXE bit set.

Where the IXE bit is set, this results in some VFP load and store operations being bounced to the support code incorrectly. However, suitable modification of the support code can detect these cases, and so mask this erratum from all users

#### **Workaround**

In the unusual event of a workaround being needed, the support code can be modified so that presentation of a VFP instruction other than a CDP, when the IXE bit is set, results in the operation being re-run.

**340351: WFE during a debug request is executed twice****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0, Fixed in r0p1.

**Description**

A WFE instruction can incorrectly cause the CPU event register to be cleared when an external debug request is received at the same time as the WFE instruction enters Ex1 pipeline stage.

**Conditions**

1. Event register of the CPU is set, ie. a wake-up action or an event occurred and no WFE instruction has been performed yet.
2. Debug is enabled and the external debug request pin (EDBGRQ) of the CPU is asserted.
3. The CPU enters debug state at the time the WFE instruction enters in Ex1 pipeline stage.
4. The debug exit is performed by scanning "SUB PC, PC, #8" through debug TAP.
5. This causes the WFE to be re-executed, but the event has already been cleared, so the processor stalls.

**Implications**

The standard WFE usage is:

Loop

```
LDREX R1, [R2]
CMP R1, #0
WFENE
BNE Loop
```

If another CPU is executing the following code sequence:

```
MOV R0, #0
STR R0, [R2]
DSB
SEV
```

The LDREX returns 1, so the WFENE is executed. Between the LDREX and WFENE, the event (SEV) from the other processor arises, so the WFENE is executed with the Event register set.

An external debug register comes in at the same time that the WFENE enters Ex1, causing the event to be cleared, and the core enters debug.

The core exits debug by scanning "SUB PC, PC, #8" through TAP, ie the WFE is re-executed and the processor is stalled.

Therefore the intervention of debug has caused the system to deadlock.

## Workaround

According to the ARM architecture, the entry into Debug should be guaranteed to leave the event set, so this scenario should not be architecturally possible. Therefore it is a bug which manifests itself only on debug.

There is a simple preventative workaround, namely that the debugger should always perform a dummy SEV prior to restarting the cpu, which is a benign operation in this situation.



**364373: Data corruption is possible if AXI memory system is not maintaining transaction order.****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0,r0p1,r0p2, Fixed in r0p3.

**Description**

There are few scenarios where MPCore can produce eviction write transactions to the same address line on each of its two master ports, so that the second eviction will start some time after the last data double-word of the first eviction has been accepted by the memory system (WREADY, WVALID and CLKEN signals high for one cycle). If the AXI memory system, or accessed AXI slave, attached to MPCore can re-order these two write transactions before the first one is actually completed (BRESP received with BVALID, BREADY and CLKEN high for one cycle), a data corruption can occur because of write-after-write hazards.

**Conditions**

1. Two or more MP11 CPUs are present, active and in SMP mode in MPCore.
2. Two or more MP11 CPUs are working on and modifying the same coherent cache line (A).
3. One linefill (B) is started on the MP11 CPU that owns the cache line. That linefill address is such that the cache line (A) will be replaced, so it is evicted (if dirty or L1 cache set as in exclusive mode).
4. One MPCore master then sends the eviction transaction address (A) on AXI address channel, and then all the burst data on the write data channels, i.e. WLAST, WVALID and WREADY are all set when the last 64-bit data value is presented on the bus.
5. No buffered response is received for that eviction transaction, and the started linefill (B) is not completed, for any reasons. Therefore, the line scheduled for eviction (A) is still in the cache.
6. A second MP11 CPU is requesting the cache line (A), and gets it through a DDI request from the MP11 CPU which has scheduled the eviction, and then allocates it in its own cache.
7. The second MP11 CPU then modifies the cache line (A) in its own cache.
8. The second MP11 CPU evicts the cache line (A) because of a cache maintenance operation or a new linefill.
9. This leads to a new eviction transaction to the same address (A) on the second MPCore AXI master port.
10. If the system is not balanced or transactions are re-ordered, the second eviction can actually complete in the accessed AXI slave before the first one.

**System design consideration**

The occurrence of this erratum depends on the system design.

Although nothing in the AXI specification prevents a memory system or a slave to have such behaviour, it is expected that few systems will have such re-ordering and hugely different response times for two transactions to the same address, or set of address, required to exhibit this erratum.

**Implications**

If AXI memory system attached to MPCore can re-order two write transactions to the same address once all data of the first transaction have been sent on the bus, data corruption can occur due to write-after-write hazards.

**Workaround**

There is no software work-around for that erratum. A hardware work-around is to check that two write transactions to the same address are treated in order by the accessed slave if the data of the first access have been accepted.

**446466: S bit value in TTBRn is not reflected on ARUSER[0] signal on page table walks****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r1p0, Fixed in r2p0.

**Description**

On page table walks, the value of the output pin ARUSERn[0] for the read transaction appearing on one of the ARM11 MPCore AXI bus master is always 1'b0, regardless of the value of the S bit (bit [1] - of Translation Table Base Register 0 and 1 (TTBR0 and TTBR1)).

TTBR0 and TTBR1 main purpose is to hold a physical base address for first-level translation table. These registers are also used to define outer cacheable attributes used for page table walking, as well as the value of the Shared attribute for the memory region accessed.

For any page table walk, use of either TTBR0 or TTBR1 depends on the virtual address and Translation Table Base Control Register (TTBCR). The Shared attribute is incorrectly ignored and the associated read transaction is always marked as being made to a non-shared memory region (ARUSERn[0] = 1'b0).

**Implications**

This erratum is visible in systems where the Shared memory attribute is used to prevent caching of data with the Outer cacheable attribute in a cache connected to the AXI interface (such as a Level 2 cache). This could be the case in a uniprocessor system or when a core is configured to be in AMP mode. In such a system, writes to translation tables marked as Shared Outer Cacheable would not be cached in the level 2 cache, and translation table walks would hit stale entries in the level 2 cache.

Reading stale entries in the level2 cache during page table walks implies that any transaction to the corresponding memory region could have incorrect virtual to physical address mapping, and/or incorrect attributes for permission checking in the MMU.

**Workaround**

Do not map TTB as outer cacheable Shared, but as non cacheable.



**498365: ETM11 prevents CPU clock stopping invoked by WFE for a CPU that it is connected to in an MPCore system****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r1p0, Fixed in r2p0.

**Description**

When an ARM11 MPCore CPU executes a WFI or WFE instruction, it waits for all transactions to complete in the L1 memory system (instruction and data side). If an ETM11 is connected, it also waits for the nETMWFIREADY signal to be de-asserted.

ETM11 de-asserts nETMWFIREADY signals after it has been notified that the CPU wants to enter IDLE state by the assertion of ETMWFIPENDING.

In the case of the execution of a WFE instruction, the ETMWFIPENDING signal is incorrectly never raised so that the ETM11 cannot de-assert the nETMWFIREADY handshake signal.

**Implications**

The implication of this erratum is that the CPU will never stop its clock when entering WFE state, and that STANDBYWFE signal will never be raised.

This had no functional implication since has the CPU will not execute further anymore instruction unless a WakeUpAction occurs (as described in ARM11 MPCore TRM DDI0360D, page 2-48). The ARM Architecture states that no assumptions should be made upon the actual execution or not of a WFE execution.

The main implication is one of power utilisation when using WFE.

**Workaround**

There is no software workaround for this erratum.

**501614: SCU access can be incorrectly denied to all CPUs through the SCU Control Register****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r1p0, Fixed in r2p0.

**Description**

the SCU Control Register can be used to control which CPU within an ARM11 MPCore Multi-processor system can write to the SCU specific registers. By default, all CPUs can modify SCU specific registers but clearing bits in the SCU Register Access Control field of the SCU Control Register has the effect of denying write access to the targeted CPU. ARM11 MPCore TRM states that there is a mechanism to prevent all bits being cleared at the same time. An error with this mechanism means that all bits can actually be cleared.

**Implications**

If all bits of SCU Register Access Control field of SCU control register are cleared, the SCU specific registers will be no longer accessible. Clearing all bits would constitute a programming error.

**Workaround**

Under normal circumstances no workaround is necessary since correctly written software should never clear all bits on the SCU Access Control Register.

**717874: No Vector Catch debug event on reset when High Vectors are enabled****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r1p0,r2p0, Fixed in r2p1.

**Description**

Vector catch is a method to generate Debug exception or to enter Debug State, triggered by the fetch of an exception vector. Based on the value of V bit of CP15 Control Register which default value out of reset is set by VINITHI input signal, the exception vector addresses sit in address range 0x00000000-0x0000001C (normal exception vectors) or in range 0xFFFF0000-0xFFFF001C (high exception vector).

A delay in the handling of VINITHI at reset in vector catch logic prevent Vector Catch on reset to be triggered when VINITHI is set to 1.

**Conditions**

The problem occurs when the following conditions are met:

1. Monitor or Halting Debug mode is selected and enabled by setting accordingly bits [15:14] of Debug Status and Control Register (DSCR, CP14 c1)
2. Vector catch on reset is enabled by setting bit 0 of Vector Catch Register (VCS, CP14 c7)
3. VINITHI input signal is set to high to select high exception vector mode out of reset
4. CPU functional reset is applied and release to initiate reset sequence.

**Implications**

Debug State entry on reset Vector Catch is not possible when high exception vectors are selected through VINITHI input signal.

**Workaround**

There are two potential workarounds for this erratum

1. When possible, use normal exception vectors so that exception vector addresses are mapped in address range 0x00000000-0x0000001C
2. Instead of setting Debug State Entry on vector catch, set a breakpoint on address 0xFFFF0000.

**721879: TLBIMVAA operation may not invalidate all required TLB entries when multiple page sizes are used.****Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r1p0,r2p0, Fixed in r2p1.

**Description**

A TLBIMVAA operation should invalidate all TLB entries with a matching MVA, regardless of the page size. Due to the erratum, the operation is only executed for entries on one single page size. So, in the case when the same MVA is present several times in the mainTLB for different page sizes, some entries are not invalidated although they should be, potentially leading to a corruption in the page table entries.

Conditions:

1. MMU is on
2. Subpage AP bits are disabled by setting XP bit in CP15 Control Register
3. The mainTLB contains at least
  - a. 1 entry for a given MVA with a given page size
  - b. another entry for the same MVA with a different page size
4. a TLBIMVAA operation is executed

Under such conditions, only 1 of the 2 entries in the TLB is invalidated, because the operation is performed on one single page size. Which page size is affected by the operation solely depends on the internal state of the processor, and is not controllable in software.

**Implications**

Some entries in the mainTLB which should be invalidated are not, thus leading to a corruption in the page table entries. This is unlikely to happen with typical Operating Systems as they are not usually using multiple page sizes with same ASID.

**Workaround**

The workaround for this erratum consist in replacing the TLBIMVAA maintenance operation by the TLBIALL operation thus invalidating all entries in the mainTLB instead of invalidating all entries with a matching MVA. This workaround, if implemented, could have a slight performance impact because it invalidates more TLB entries than required.



## Errata - Documentation

### **341802: TRM must state that BTAC is not automatically flushed when FCSE PID or Context ID registers are changed**

#### **Status**

Affects: product MPCore.

Fault status: Doc, Present in: r0p1, Fixed in r0p2.

#### **Description**

MPCore MP11 CPUs differ from other ARM11 family CPUs in the way Context ID and FCSE PID changes are handled. Though the ARM Architecture revF does not specify it, the BTAC of ARM1136, ARM1156 and ARM1176 is automatically flushed when the Context ID or FCSE PID is changed by a CP15 write access. The MP11 in MPCore does not automatically flush the BTAC when either of these are written.

Any application or Operating System relying on the fact, incidentally or not, that the BTAC is automatically flushed could then stop working reliably on MPCore compared to other ARM11 family CPUs.

The MPCore TRM must therefore make clear that MP11 BTACs are not automatically flushed on a Context ID or FCSE PID change, and if this is required, the software must explicitly flush the BTAC by executing a "Flush Entire Branch Target Cache" CP15 instruction.

#### **Implications**

none

#### **Workaround**

none



## Errata – Driver Software

**There are no Errata in this Category**

## Errata – Implementation

### **447678: SCU duplicated TAG RAM BIST data is presented in incorrect order**

#### **Status**

Affects: product MPCore.

Fault status: Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r1p0, Fixed in r2p0.

#### **Description**

According to the ARM11 MPCore TRM, when SCU duplicated TAG RAMs are accessed by setting MBISTCE[16] or MBISTCE[17], MBISTDOUT bus should reflect RAM arrays data as follow:

1. MBISTDOUT[31:0] represent CPU0 duplicated TAG RAM for way0, way2 respectively.
2. MBISTDOUT[63:32] represent CPU0 duplicated TAG RAM for way1, way3 respectively.
3. MBISTDOUT[95:64] represent CPU1 duplicated TAG RAM for way0, way2 respectively.
4. MBISTDOUT[127:96] represent CPU1 duplicated TAG RAM for way1, way3 respectively.
5. MBISTDOUT[159:128] represent CPU2 duplicated TAG RAM for way0, way2 respectively.
6. MBISTDOUT[191:160] represent CPU2 duplicated TAG RAM for way1, way3 respectively.
7. MBISTDOUT[223:192] represent CPU3 duplicated TAG RAM for way0, way2 respectively.
8. MBISTDOUT[256:224] represent CPU3 duplicated TAG RAM for way1, way3 respectively.

Due to this erratum, the RAM array data for CPU1, CPU2 and CPU3 (if present) are swapped on the MBISTDOUT bus in the following manner:

1. MBISTDOUT[95:64] represent CPU1 duplicated TAG RAM for way1, way3 respectively.
2. MBISTDOUT[127:96] represent CPU1 duplicated TAG RAM for way0, way2 respectively.
3. MBISTDOUT[159:128] represent CPU2 duplicated TAG RAM for way1, way3 respectively.
4. MBISTDOUT[191:160] represent CPU2 duplicated TAG RAM for way0, way2 respectively.
5. MBISTDOUT[223:192] represent CPU3 duplicated TAG RAM for way1, way3 respectively.
6. MBISTDOUT[256:224] represent CPU3 duplicated TAG RAM for way0, way2 respectively.

#### **Implications**

This erratum can lead to incorrect fault reporting during manufacturing test:

1. If CPU1, CPU2 or CPU3 are present and ways 0 and 1, and ways 2 and 3 are written to through the MBIST interface with different values, both RAM arrays representing those ways will be incorrectly reported as broken.
2. If ways 0 and 1, and ways 2 and 3 are written to with the same value and one array only is faulty, the fault will be incorrectly reported for the wrong array.

## Workaround

There is no software workaround for this erratum.

Only hardware work-around is possible and would imply swapping groups of 32 bits of MBISTDATAOUT bus, when SCU duplicated TAG RAMS are selected (MBISTCE[16] or MBISTCE[17] set), in the following manner between ARM11 MPCore and the external BIST controller:

1. Swap MBISTDATAOUT[127:96] with MBISTDATOUT[95:64]
2. Swap MBISTDATAOUT[191:160] with MBISTDATOUT[159:128]
3. Swap MBISTDATAOUT[255:224] with MBISTDATOUT[223:192]