ARM Core

ARM1156T2F-S™ / ARM1156T2-S™ (AT360 / AT370) **Errata Notice**

This document contains all errata known at the date of issue in supported releases up to and including revision r0p4 of ARM1156T2F-S(AT360) and ARM1156T2-S (AT370) products.

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:support-cores@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 **14**

There are no Errata in this Category **14**

ERRATA - CATEGORY 2 **15**

350026: Cache Debug Control Register (p15,7,c15,c0,0) reads as zero **15**

352098: The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data **16**

374859: IT state bits not cleared on imprecise abort in debug state **18**

374957: VFP and GCP instructions are not executed in debug state when the T bit is set **20**

375900: Disabling the instruction cache during a linefill can corrupt instructions or result in deadlock **21**

376234: DTCM stall during a store instruction stalled in the pipeline may cause corrupted data or system deadlock **23**

377047: Multiple occurring MPU aborts may result in the wrong IFSR value **26**

377050: IFSR value incorrect after parity error followed by an MPU abort **28**

377509: A PLD instruction which causes a background fault may deadlock **29**

377882: Interrupts from VIC interface may be taken multiple times **30**

379894: Load Multiple to the PC can be corrupted or result in deadlock **32**

388230: Cache clean when data cache disabled can writeback to the wrong address **34**

393694: Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration **35**

393696: DTCM stalls may cause data corruption **37**

393697: Data cache dirty RAM parity error on a write through region of memory can write back stale data **39**

393698: VFP or GCP instruction that stalls may not trace its data with the ETM **40**

395146: VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction **41**

397735: Write back data cache entry evicted by write through entry causes data corruption **43**

397751: Data cache clean operations may not abort on tag or data RAM parity errors **44**

398952: Tag or valid parity error on a load multiple to the PC can deadlock **46**

414935: Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill **47**

720619: Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line **50**

720627: Invalidate Instruction Cache operations can fail 52

ERRATA - CATEGORY 3 54

333141: VFP can take an inexact exception on non-CDP VFP instructions 54

374844: STREX to ITCM that fails its condition codes immediately followed by a STM to SO/DEVNS memory may cause a deadlock 55

376081: ITCM error caused by data access is not counted by performance monitor 56

376232: DTCM error counted multiple times by performance monitor whilst DTCM stalled. 57

378833: Performance monitor does not count instruction cache errors when set to reload on error 58

380083: Enabling the instruction cache during an external fetch may result in duplicate cache entries 59

389801: Cache clean after reprogramming the MPU may cause subsequent instructions to fail 61

391562: Vector catch on IRQ may not trigger when the VIC is enabled 62

391890: Interrupted folded branch may be missed from ETM trace 63

393338: A prefetch after an aborted prefetch may be ignored 64

393703: Conditional load or store instructions executed after reset may deadlock 66

394384: Invalidate data cache line by address does not cause an abort on a valid error 67

394634: Return stack events not counted correctly by performance monitor 68

396716: Data cache clean instructions with valid RAM parity errors write to external memory 69

397753: Multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory 70

401148: ETM may not gain data synchronization on power up 71

434752: FAR/FSR write immediately following precise abort can corrupt FAR/FSR 73

497917: Wait For Interrupt can cause deadlock when FREEDBGTCKEN is HIGH 74

ERRATA - IMPLEMENTATION 75

381511: Pipelined reset may cause ATPG DRC errors 75

391574: RREADY AXI signals change when ACLKEN is low 76

ERRATA - TESTCHIP 77

352364: ARM1156 MBIST test can fail if multiple arrays are simultaneously enabled 77

397822: ARM1156 testchip has incorrect JTAG ID code / part number 78

Introduction

Scope

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

- a unique defect tracking identifier
- 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.
Implementation	Errata that are of particular interest to those implementing the product and that have no software implications
Testchip	Errata that are identified with components in the Testchip RTL that make up the validation testbench such as the MBIST Controller and AXI components.

Note about ARM1136 Errata

Any workarounds introduced in ARM1136 systems are not required when that system is ported to use an ARM1156 core. Unless they explicitly appear in this errata document, all existing ARM1136 r0p2 errata have been fixed for the r0p0 release of the ARM1156 core.

Change Control

04 Aug 2009: Changes in Document v11

Page	Status	ID	Cat	Summary
50	New	720619	Cat 2	Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line
52	New	720627	Cat 2	Invalidate Instruction Cache operations can fail

04 Feb 2008: Changes in Document v10

Page	Status	ID	Cat	Summary
74	New	497917	Cat 3	Wait For Interrupt can cause deadlock when FREEDBGTCKEN is HIGH

25 May 2007: Changes in Document v9

Page	Status	ID	Cat	Summary
71	Updated	401148	Cat 3	ETM may not gain data synchronization on power up
73	New	434752	Cat 3	FAR/FSR write immediately following precise abort can corrupt FAR/FSR

08 Mar 2007: Changes in Document v8

Page	Status	ID	Cat	Summary
47	Updated	414935	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill
63	Updated	391890	Cat 3	Interrupted folded branch may be missed from ETM trace
71	New	401148	Cat 3	ETM may not gain data synchronization on power up

29 Jan 2007: Changes in Document v7

Page	Status	ID	Cat	Summary
23	Updated	376234	Cat 2	DTCM stall during a store instruction stalled in the pipeline may cause corrupted data or system deadlock
37	Updated	393696	Cat 2	DTCM stalls may cause data corruption
47	Updated	414935	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill

12 Dec 2006: Changes in Document v6

Page	Status	ID	Cat	Summary
47	New	414935	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill

28 Jul 2006: Changes in Document v5

Page	Status	ID	Cat	Summary
15	Updated	350026	Cat 2	Cache Debug Control Register (p15,7,c15,c0,0) reads as zero

16	Updated	352098	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data
18	Updated	374859	Cat 2	IT state bits not cleared on imprecise abort in debug state
20	Updated	374957	Cat 2	VFP and GCP instructions are not executed in debug state when the T bit is set
21	Updated	375900	Cat 2	Disabling the instruction cache during a linefill can corrupt instructions or result in deadlock
23	Updated	376234	Cat 2	DTCM stall during a store instruction stalled in the pipeline may cause corrupted data or system deadlock
26	Updated	377047	Cat 2	Multiple occurring MPU aborts may result in the wrong IFSR value
28	Updated	377050	Cat 2	IFSR value incorrect after parity error followed by an MPU abort
29	Updated	377509	Cat 2	A PLD instruction which causes a background fault may deadlock
30	Updated	377882	Cat 2	Interrupts from VIC interface may be taken multiple times
34	Updated	388230	Cat 2	Cache clean when data cache disabled can writeback to the wrong address
35	New	393694	Cat 2	Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration
37	New	393696	Cat 2	DTCM stalls may cause data corruption
39	New	393697	Cat 2	Data cache dirty RAM parity error on a write through region of memory can write back stale data
40	New	393698	Cat 2	VFP or GCP instruction that stalls may not trace its data with the ETM
41	New	395146	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction
43	New	397735	Cat 2	Write back data cache entry evicted by write through entry causes data corruption
44	New	397751	Cat 2	Data cache clean operations may not abort on tag or data RAM parity errors
46	New	398952	Cat 2	Tag or valid parity error on a load multiple to the PC can deadlock
55	Updated	374844	Cat 3	STREX to ITCM that fails its condition codes immediately followed by a STM to SO/DEVNS memory may cause a deadlock
56	Updated	376081	Cat 3	ITCM error caused by data access is not counted by performance monitor
57	Updated	376232	Cat 3	DTCM error counted multiple times by performance monitor whilst DTCM stalled.
58	Updated	378833	Cat 3	Performance monitor does not count instruction cache errors when set to reload on error
59	Updated	380083	Cat 3	Enabling the instruction cache during an external fetch may result in duplicate cache entries
61	Updated	389801	Cat 3	Cache clean after reprogramming the MPU may cause subsequent instructions to fail
62	Updated	391562	Cat 3	Vector catch on IRQ may not trigger when the VIC is enabled
63	Updated	391890	Cat 3	Interrupted folded branch may be missed from ETM trace
64	New	393338	Cat 3	A prefetch after an aborted prefetch may be ignored

66	New	393703	Cat 3	Conditional load or store instructions executed after reset may deadlock
67	New	394384	Cat 3	Invalidate data cache line by address does not cause an abort on a valid error
68	New	394634	Cat 3	Return stack events not counted correctly by performance monitor
69	New	396716	Cat 3	Data cache clean instructions with valid RAM parity errors write to external memory
70	New	397753	Cat 3	Multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory
75	Updated	381511	IMPL	Pipelined reset may cause ATPG DRC errors
76	Updated	391574	IMPL	RREADY AXI signals change when ACLKEN is low
78	New	397822	Testchip	ARM1156 testchip has incorrect JTAG ID code / part number
77	Updated	352364	Testchip	ARM1156 MBIST test can fail if multiple arrays are simultaneously enabled

15 Jun 2006: Changes in Document v4

Page	Status	ID	Cat	Summary
18	New	374859	Cat 2	IT state bits not cleared on imprecise abort in debug state
20	New	374957	Cat 2	VFP and GCP instructions are not executed in debug state when the T bit is set
21	New	375900	Cat 2	Disabling the instruction cache during a linefill can corrupt instructions or result in deadlock
23	New	376234	Cat 2	DTCM stall during a store instruction stalled in the pipeline may cause corrupted data or system deadlock
26	New	377047	Cat 2	Multiple occurring MPU aborts may result in the wrong IFSR value
28	New	377050	Cat 2	IFSR value incorrect after parity error followed by an MPU abort
29	New	377509	Cat 2	A PLD instruction which causes a background fault may deadlock
30	New	377882	Cat 2	Interrupts from VIC interface may be taken multiple times
32	Updated	379894	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock
34	New	388230	Cat 2	Cache clean when data cache disabled can writeback to the wrong address
55	New	374844	Cat 3	STREX to ITCM that fails its condition codes immediately followed by a STM to SO/DEVNS memory may cause a deadlock
56	New	376081	Cat 3	ITCM error caused by data access is not counted by performance monitor
58	New	378833	Cat 3	Performance monitor does not count instruction cache errors when set to reload on error
57	New	376232	Cat 3	DTCM error counted multiple times by performance monitor whilst DTCM stalled.
59	New	380083	Cat 3	Enabling the instruction cache during an external fetch may result in duplicate cache entries
61	New	389801	Cat 3	Cache clean after reprogramming the MPU may cause subsequent instructions to fail
62	New	391562	Cat 3	Vector catch on IRQ may not trigger when the VIC is enabled
63	New	391890	Cat 3	Interrupted folded branch may be missed from ETM trace

76	New	391574	IMPL	RREADY AXI signals change when ACLKEN is low
75	New	381511	IMPL	Pipelined reset may cause ATPG DRC errors
77	New	352364	Testchip	ARM1156 MBIST test can fail if multiple arrays are simultaneously enabled

12 Apr 2006: Changes in Document v3

Page	Status	ID	Cat	Summary
32	New	379894	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock

1517 Jun 2005: Changes in Document v2

Page	Status	ID	Cat	Summary
16	New	352098	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data
15	New	350026	Cat 2	Cache Debug Control Register (p15,7,c15,c0,0) reads as zero

10 Mar 2005: Changes in Document v1

Page	Status	ID	Cat	Summary
10	New	333141	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	r0p4
352364	Testchip	ARM1156 MBIST test can fail if multiple arrays are simultaneously enabled	X	X			
381511	IMPL	Pipelined reset may cause ATPG DRC errors	X	X			
391574	IMPL	RREADY AXI signals change when ACLKEN is low	X	X			
397822	Testchip	ARM1156 testchip has incorrect JTAG ID code / part number	X	X			
350026	Cat 2	Cache Debug Control Register (p15,7,c15,c0,0) reads as zero	X	X			
352098	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data	X	X			
374859	Cat 2	IT state bits not cleared on imprecise abort in debug state	X	X			
374957	Cat 2	VFP and GCP instructions are not executed in debug state when the T bit is set	X	X			
375900	Cat 2	Disabling the instruction cache during a linefill can corrupt instructions or result in deadlock	X	X			
376234	Cat 2	DTCM stall during a store instruction stalled in the pipeline may cause corrupted data or system deadlock	X	X			
377047	Cat 2	Multiple occurring MPU aborts may result in the wrong IFSR value	X	X			
377050	Cat 2	IFSR value incorrect after parity error followed by an MPU abort	X	X			
377509	Cat 2	A PLD instruction which causes a background fault may deadlock	X	X			
377882	Cat 2	Interrupts from VIC interface may be taken multiple times	X	X			
379894	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock	X				
388230	Cat 2	Cache clean when data cache disabled can writeback to the wrong address	X	X			
393694	Cat 2	Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration	X	X			
393696	Cat 2	DTCM stalls may cause data corruption	X	X			
393697	Cat 2	Data cache dirty RAM parity error on a write through region of memory can write back stale data	X	X			

ID	Cat	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r0p4
393698	Cat 2	VFP or GCP instruction that stalls may not trace its data with the ETM	X	X			
395146	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction	X	X			
397735	Cat 2	Write back data cache entry evicted by write through entry causes data corruption	X	X			
397751	Cat 2	Data cache clean operations may not abort on tag or data RAM parity errors	X	X			
398952	Cat 2	Tag or valid parity error on a load multiple to the PC can deadlock	X	X			
414935	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill	X	X	X		
720619	Cat 2	Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line	X	X	X	X	X
720627	Cat 2	Invalidate Instruction Cache operations can fail	X	X	X	X	X
333141	Cat 3	VFP can take an inexact exception on non-CDP VFP instructions	X	X	X	X	X
374844	Cat 3	STREX to ITCM that fails its condition codes immediately followed by a STM to SO/DEVNS memory may cause a deadlock	X	X			
376081	Cat 3	ITCM error caused by data access is not counted by performance monitor	X	X			
376232	Cat 3	DTCM error counted multiple times by performance monitor whilst DTCM stalled.	X	X			
378833	Cat 3	Performance monitor does not count instruction cache errors when set to reload on error	X	X			
380083	Cat 3	Enabling the instruction cache during an external fetch may result in duplicate cache entries	X	X			
389801	Cat 3	Cache clean after reprogramming the MPU may cause subsequent instructions to fail	X	X			
391562	Cat 3	Vector catch on IRQ may not trigger when the VIC is enabled	X	X			
391890	Cat 3	Interrupted folded branch may be missed from ETM trace	X	X	X	X	X
393338	Cat 3	A prefetch after an aborted prefetch may be ignored	X	X			
393703	Cat 3	Conditional load or store instructions executed after reset may deadlock	X	X			
394384	Cat 3	Invalidate data cache line by address does not cause an abort on a valid error	X	X			

ID	Cat	Summary of Erratum	r0p0	r0p1	r0p2	r0p3	r0p4
394634	Cat 3	Return stack events not counted correctly by performance monitor	X	X			
396716	Cat 3	Data cache clean instructions with valid RAM parity errors write to external memory	X	X			
397753	Cat 3	Multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory	X	X			
401148	Cat 3	ETM may not gain data synchronization on power up	X	X	X	X	
434752	Cat 3	FAR/FSR write immediately following precise abort can corrupt FAR/FSR	X	X	X	X	
497917	Cat 3	Wait For Interrupt can cause deadlock when FREEDBGTCKEN is HIGH	X	X	X	X	X

Errata - Category 1

There are no Errata in this Category

Errata - Category 2

350026: Cache Debug Control Register (p15,7,c15,c0,0) reads as zero

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The Cache Debug Control Register is used to over-ride the cache behaviour defined by the ARM1156 default memory map or as programmed into the Memory Protection Unit (MPU). The Cache Debug Control Register controls the following behaviour

- Force write back cacheable regions to behave as write through
- Disable Instruction linefills
- Disable Data cache linefills

When reading this register, all bits are returned as zero, irrespective of the value programmed into the register.

Writes to this register function correctly.

Implications

Read modify write operations performed on this register will result in the default mode of operation being restored. For example, if an application used this register to force write through behaviour for all data cached accesses, subsequent reads of the cache debug control register would not correctly report this (for example a read performed by a debugger). This would result in this register being cleared as a result of a write, and cached accesses being treated as write-through or write-back depending on the MPU or default memory map.

Workaround

The Cache Debug Control register should only be used for debug operations. Application code should not make use of this register.

352098: The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data

Status

Affects:	product(s) ARM1156T2F-S.
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

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 multiples 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.

374859: IT state bits not cleared on imprecise abort in debug state

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

When an imprecise abort occurs in debug state the CPSR should be set as for a normal imprecise abort exception entry. This includes setting bits 26:25 and 15:10 (the IT state bits) to zero, so that the exception handler is not affected by any IT instruction in progress at the time of the abort.

This erratum occurs if an imprecise abort is taken whilst the processor is in debug state. In this case, the IT state bits are not set to zero in the CPSR if the exception handler is to be executed in Thumb state.

Conditions

1. The TE bit in the CP15 control register is set
2. The processor is in debug state
3. The IT bits in the CPSR are non-zero
4. An imprecise data abort occurs

Implications

If the imprecise abort occurs before the CPSR is read by the debugger, then it will have the IT state bits incorrectly set. If the debugger subsequently writes this incorrect value back to the CPSR then the IT state bits will be incorrect when the processor leaves debug state.

If an imprecise abort occurs after the CPSR has been written to by the debugger, but before the processor leaves debug state, then the IT state bits will be incorrect when the processor leaves debug state.

In either of these cases, when the processor is restarted the IT state bits will get applied to the subsequent instructions executed. Therefore between 1 and 4 of the following instructions may not be executed, depending on the state of the flags in the CPSR.

Workaround

The debugger should normally write to the PC and CPSR before returning from debug state, otherwise the processor's behaviour is UNPREDICTABLE. It should also already be executing a data synchronisation barrier before examining any processor state. The debugger can determine if an imprecise abort has occurred whilst in debug state by examining the DSCR[7] sticky imprecise abort bit. To work around the erratum, the debugger should check the DSCR[7] bit and if it is set then it should set the IT state bits to zero when it writes to the CPSR. Between writing the CPSR and restarting the processor, the debugger should not execute any further instructions that could cause an imprecise abort to occur.

The following pseudo-code shows an example of how a debugger can work around the erratum:

```
MRC p15,0,rd,c7,c10,0  ; DSB
Read DSCR
Read and store CPSR
if DSCR[7] set then stored_CPSR &= 0xF9FF03FF
Execute any other instructions needed in debug state
Write stored_CPSR to CPSR
restart
```

374957: VFP and GCP instructions are not executed in debug state when the T bit is set

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The ACPINSTRV signal is an output from the ARM1156 that indicates to an external coprocessor when a new instruction reaches the decode stage of the pipeline. If the instruction is a coprocessor instruction, then the VFP or external coprocessor must decode the instruction and indicate to the processor whether it has accepted or bounced the instruction.

If a VFP or GCP coprocessor instruction is executed while the processor is in debug state and the T bit in the CPSR is set, then the ACPINSTRV signal does not get correctly asserted for the coprocessor instruction. The coprocessor is not aware that an instruction is available, and therefore will not decode it. The processor will then stall indefinitely waiting for the coprocessor to respond.

Conditions

1. The core is in debug state
2. The T bit in the CPSR is set
3. A GCP or VFP coprocessor instruction is executed

Implications

The erratum will only occur if the debugger tries to execute a VFP or GCP coprocessor instruction. If it occurs the processor will not execute the coprocessor instruction, and will prevent further instructions from executing.

Workaround

If the debugger does not execute any VFP or GCP coprocessor instructions then no workaround is necessary. If the debugger does need to execute any such coprocessor instructions then it should first write to the CPSR to ensure that the T bit is not set. The debugger can restore the previous value of the T bit if necessary after the coprocessor instructions have been executed.

375900: Disabling the instruction cache during a linefill can corrupt instructions or result in deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The instruction cache can be enabled or disabled at any time by writing to bit 12 of the CP15 control register. If the instruction cache is disabled while a linefill is in operation there is a possibility of one of two erroneous behaviours:

1. Incorrect instructions are passed to the core for execution.
2. The linefill never completes resulting in a deadlock.

Conditions

1. The instruction cache is enabled.
2. The instruction cache is disabled by writing to the CP15 control register.
3. At the time the CP15 write is performed a new instruction cache linefill is started.

In addition to the above conditions, specific timing between internal signals is also required to stimulate this erratum. Therefore, any code that replicates the given conditions may not stimulate this erratum.

Implications

Typically, once the instruction cache is turned on, it is rarely turned off again. Compilers cannot generate code to enable or disable the instruction cache.

If this erratum does occur, it will result in one of two possible outcomes.

1. The instructions being fetched by the linefill will become corrupted. The processor will then execute these incorrect instructions.
2. The processor will deadlock and will not execute any further instructions.

Workaround

This erratum can be worked around by ensuring that there are no linefills occurring at the point that the instruction cache is disabled. The following code sequence shows how to disable the cache without causing any linefills. The code should be placed either in a region of memory that is marked as non-cacheable, or in the ITCM. The twelve NOPs ensure that following code cannot start a new linefill.

```
MCR p15, 0, R0, c7, c5, 4 ; Flush prefetch buffer
MRC p15, 0, R0, c1, c0, 0 ; Read control register
BIC R0, R0, #0x1000       ; Clear bit 12
MCR p15, 0, R0, c1, c0, 0 ; Write control register, disabling the instruction cache
NOP
NOP
NOP
```

```
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

376234: DTCM stall during a store instruction stalled in the pipeline may cause corrupted data or system deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

A store instruction that gets stalled in the pipeline for exactly 2 cycles may behave incorrectly if it occurs at the same time as a stall caused by an earlier store instruction that resulted in the assertion of the nDTCDATARDY signal on the DTCM interface.

There are two sets of conditions that can cause the erratum. The first applies to both the ARM1156T2-S and ARM1156T2F-S, while the second only applies to the ARM1156T2F-S.

Conditions

Condition set 1:

1. One to three store instructions are executed, which fill the cache write buffer. The first of these is to DTCM and must cause either an error correction cycle or a wait cycle from the DTCM.
2. An instruction is executed that sets the flags in the CPSR.
3. A conditional load from the DTCM is executed and it fails its condition code. This load must cause a parity error response from the DTCM, but it will not cause an abort due to the condition code failure.
4. There must be either zero or one instructions between the flag setting instruction and the conditional load. If there is an instruction, it must take a single cycle to execute and cannot be a load or store instruction.
5. A saturation instruction is executed. This must be one of the following instructions: USAT, SSAT, USAT16, SSAT16, QADD, QSUB, QDADD, QDSUB, QADD8, QADD16, QSUB8, QSUB16, SHADD8, SHADD16, SHSUB8, SHSUB16, UQADD8, UQADD16, UQSUB8, UQSUB16, UHADD8, UHADD16, UHSUB8, UHSUB16, QADDSUBX, QSUBADDX, SHADDSUBX, SHSUBADDX, UQADDSUBX, UQSUBADDX, UHADDSUBX, UHSUBADDX.
6. A store instruction is executed which uses the destination register of the saturate as its base address. This can be either a double word aligned store multiple of at least 3 registers, a double word unaligned store multiple of 2 registers, a doubleword unaligned store double, or an unaligned store word or halfword. Alternatively, it could be two separate store instructions with the base address of the first dependent on the saturate instruction.
7. In addition to the above code sequence it is also required that when the DTCM stalls, it is only for one clock cycle.

OR

Condition set 2:

1. One to three store instructions are executed, which fill the cache write buffer. The first of these is to DTCM and must cause either an error correction cycle or a wait cycle from the DTCM.

2. A long VFP operation is in progress. This could be an operation that takes a long time, such as a divide, or a vector of shorter operations.
3. An instruction is executed that sets the flags in the CPSR.
4. A conditional load from the DTCM is executed and it fails its condition code. This load must cause a parity error response from the DTCM, but it will not cause an abort due to the condition code failure.
5. There must be either zero or one instructions between the flag setting instruction and the conditional load. If there is an instruction, it must take a single cycle to execute and cannot be a load or store instruction.
6. A VFP CDP instruction is executed. This must be dependant on the result of the earlier VFP operation. The earlier operation must finish 2 cycles after the CDP is executed, so that the CDP stalls the pipeline due to the interlock for exactly 2 cycles.
7. A store instruction is executed. This can be either a double word aligned store multiple of at least 3 registers, a double word unaligned store multiple of 2 registers, a doubleword unaligned store double, or an unaligned store word or halfword. Alternatively, it could be two separate store instructions back to back.
8. In addition to the above code sequence it is also required that when the DTCM stalls, it is only for one clock cycle.

In addition to the above conditions, specific timing between internal signals is also required to stimulate this erratum. Therefore, any code that replicates the given conditions may not stimulate this erratum.

Implications

This erratum requires one of two specific sequences of instructions to occur at the same time as stalls or errors are occurring on the DTCM. The code sequence for condition set 1 requires the result of a saturate instruction to be used as an address for a following store instruction. Such a sequence is unlikely to be of any use, and therefore is unlikely to exist in real code. It is not code that would be generated by a compiler. The code sequence for condition set 2 is something that a compiler could potentially generate, although it is only valid for the ARM1156T2F-S.

DTCM stalls and errors are rare events, and so the combination of one of the code sequences at the same time as the stalls and errors is likely to be very rare. If nDTCDATARDY is not used, this erratum can not occur.

This erratum may cause any following load or store instructions to be executed out of program order. Possible effects of this are:

1. Data corruption if the order of any stores and loads are swapped.
2. Deadlock.
3. No effect at all. In some situations there may be no visible effects if operations are out of order. If there are no load or store operations immediately following these conditions then the internal buffers in ARM1156 will empty and execution will return to normal.

Workaround

For the ARM1156T2-S processor, only condition set 1 applies. If any data processing instruction is placed between the saturate instruction and the store instruction that depends on it then the erratum will not occur.

For the ARM1156T2F-S processor, both condition sets apply. Condition set 1 can be worked around in the same way as for the ARM1156T2-S. There is no workaround for condition set 2.

377047: Multiple occurring MPU aborts may result in the wrong IFSR value

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Speculative instruction fetches are often issued by the ARM1156T2(F)-S in order to improve performance. Sometimes these speculative fetches are discarded and any abort associated with the discarded fetch is ignored. However, under the conditions described in this erratum, an MPU fault caused by a discarded speculative fetch can cause a subsequent MPU abort to report the wrong IFSR value.

Conditions

1. The processor is fetching instructions speculatively.
2. The speculative instruction fetch generates an MPU fault. This can be either a background fault or a permission fault.
3. No further instructions are fetched.
4. The processor determines that the speculatively fetched instructions are not required, for example due to a branch that was mispredicted. This results in the correct instructions being fetched from a different address.
5. The first instruction fetched from this new address causes an MPU permission fault.

The IFSR value reported will be that for a background fault not a permission fault.

In addition to the above conditions, specific timing between internal signals is also required to stimulate this erratum. Therefore, any code that replicates the given conditions may not stimulate this erratum.

Implications

This erratum only applies if the second MPU abort is a permission fault. If the fetch due to condition (4) causes a background fault then the erratum does not apply. Similarly, aborts due to RAM errors or Level 2 errors outside the ARM1156T2(F)-S are not affected by this erratum. All types of abort occur as expected, only the IFSR value is affected by this erratum.

It is expected that MPU aborts on an ARM1156T2(F)-S are likely to be treated as fatal. The software abort handler is not expected to fix the cause of the abort and then try to continue executing instructions. Therefore the IFSR value being incorrect is unlikely to affect the operation of the system.

Workaround

This erratum can be worked around by checking the IFSR and IFAR values for consistency after an MPU abort occurs. If a background fault appears to occur in a privileged region then the IFSR value should be disregarded.

Alternatively, the abort handler can re-execute the instruction that caused the abort. The instruction will abort again, but the IFSR register will contain the correct value this time.

377050: IFSR value incorrect after parity error followed by an MPU abort

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Pipelining in the Instruction side memory system can cause multiple memory system errors or faults to occur in quick succession. Normally only the first causes an abort and later fetches being processed are discarded. In this erratum a parity error followed immediately by a MPU fault results in an abort with the wrong IFSR value.

Conditions

1. The PE bit in the Auxiliary Control Register is set to enable parity errors.
2. The Prefetch Unit is issuing fetches one per cycle for at least two cycles.
3. The two fetches must either be both to TCM or both to the instruction cache.
4. If the fetches are to instruction cache, the IR bit in the CP15 Auxiliary Control Register must not be set.
5. The first fetch results in a parity error.
6. The next fetch, which must be sequential, finishes exactly one cycle later and causes a MPU fault, either a background fault or a permission fault.

The IFSR value reported for the parity error will be that for the MPU fault.

Implications

This erratum only applies for single cycle fetches which are either:

1. Fetches which hit in the cache, or
2. Fetches which hit the Instruction TCM and for which there are no waits.

Aborts occur as expected, it is only the IFSR which is affected.

This erratum only applies if instruction fetching is sequential and occurs past the end of an MPU region, which should not normally occur. A branch to a memory location which causes a MPU fault will not be affected by this erratum.

Workaround

To avoid fetching past the end of a MPU region no instruction data should be placed at the last 64 bit word in an MPU region. This will stop this erratum occurring.

Alternatively, the abort handler can re-execute the instruction that caused the abort. The instruction will abort again, but the IFSR register will contain the correct value this time.

377509: A PLD instruction which causes a background fault may deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

If an instruction tries to access a memory address for which there is no MPU region defined, it will generate a background fault. For normal load and store instructions a background fault will cause a data abort to occur. PLD instructions cannot cause a data abort, therefore a background fault should not have any effect on the PLD.

However, under some conditions, if the PLD generates a background fault it may cause the processor to deadlock.

Conditions

1. Bit 21 of the CP15 control register is set to enable low interrupt latency configuration.
2. A load or store instruction is executed which does not hit in the cache or TCM, and so causes an external fetch from the level two memory system.
3. A PLD instruction is executed which causes a background fault. This must happen whilst the previous external fetch is still in progress.

Implications

A PLD instruction would not normally be done to an address without an MPU region defined, as such data could not be accessed by a normal load instruction.

If the erratum does occur, it will cause the processor to deadlock, and not execute any further instructions.

Workaround

Several workarounds are available:

1. Ensure all PLD instructions are to addresses that have a defined memory region.
2. Define MPU region 0 to cover the entire address space. The access permissions on this region can be set to no access to provide equivalent functionality to the background region. If an instruction accesses this region it will get a permission fault rather than an access fault, and therefore will not be affected by the erratum.
3. Precede the PLD instruction with a Data Memory Barrier instruction, to ensure that there are no outstanding external fetches taking place.
4. Disable low interrupt latency configuration by clearing bit 21 in the CP15 control register.

377882: Interrupts from VIC interface may be taken multiple times

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The VIC interface specification indicates that while IRQADDRV is asserted, the processor should mask the nIRQ interrupt input. The ARM Vectored Interrupt Controller (PL192), or some other Vectored Interrupt Controller, can be used to take advantage of this because it masks the taken interrupt when IRQADDRV is deasserted. This means that, once a VIC handshake has taken place for a particular interrupt (and therefore IRQ exception entry has occurred) the processor and the VIC work together to ensure that the same interrupt cannot be taken again. Therefore the programmer can re-enable IRQ interrupts (by clearing the CPSR I-bit) without having first cleared the interrupt source, and this leads to lower interrupt latency for IRQ interrupts that interrupt the handling of lower priority IRQ interrupts.

The ARM1156 does not correctly mask the nIRQ interrupt in this way, which means that, after IRQACK has been deasserted, but before IRQADDRV has been deasserted, there is potential for the processor to perform exception entry, stack the necessary registers, re-enable IRQ interrupts, and take another exception for the same interrupt, before the VIC has masked the interrupt.

Conditions

1. The processor is using the VIC interface (enabled by CP15 Register 1 Control Register, bit [24]).
2. The programmer re-enables interrupts by clearing the CPSR I bit before clearing the interrupt source.
3. The Vectored Interrupt Controller connected to the VIC interface takes more cycles to mask the interrupt and deassert IRQADDRV than the processor takes to perform exception entry, stack the necessary registers and re-enable interrupts. Typically this will occur when the exception entry sequence is particularly quick, and the Interrupt Controller is particularly slow.

The exact conditions are a function of the timing of the system. The Interrupt Controller may take a large number of processor clock cycles to mask the interrupt if it is being clocked at a slower rate to the processor, and especially if the VIC interface is being used asynchronously.

The processor may take fewer cycles to perform exception entry if the instructions and data used in exception entry are in fast memory like the cache or a TCM, if the number of registers pushed to the processor stack is kept to a minimum (r14 and SPSR_irq must be stacked) and if the I-bit is cleared within a small number of instructions.

Implications

Most system software which uses the Vectored Interrupt Controller does not clear the CPSR Interrupt mask until the source interrupt has been cleared. This software does not make use of the ability of the Vectored Interrupt Controller to mask an interrupt once it has been taken, but before the source of the interrupt is cleared. These systems suffer no implications from this erratum.

For software that does make use of the ability of the Vectored Interrupt Controller to mask an interrupt once it has been taken, but before the source of the interrupt is cleared, then:

A single IRQ interrupt presented on the VIC interface may cause multiple IRQ exceptions to be taken in systems where the hardware masking of the nIRQ signal in the interrupt controller is taking a large number of processor clock cycles. The exact number of cycles is a function of the initial code within the interrupt service routine.

With the PL192, if IRQACK is presented when nIRQ and IRQADDRV are de-asserted, then the PL192 will assert nIRQ again. This in turn can lead to a new interrupt entry within the ARM1156T2(F)-S, so potentially resulting in an indefinite number of interrupt entries. This is effectively a livelock, though as the stack will be incrementing, at some point a stack overflow will occur; this is likely to be a fatal error.

Workaround

1. No error will occur if the program does not make use of the ability of the Vectored Interrupt Controller to mask interrupts once they have been taken, but before they are cleared. Therefore, one possible workaround is to write the interrupt handler to clear the interrupt source before clearing the CPSR interrupt mask.
2. A second workaround is to write the interrupt handler to behave in the same way regardless of the number of times it is called. This workaround is not suitable for use with the PL192 and any other interrupt controller which re-asserts nIRQ if IRQACK is asserted.
3. A third workaround is to ensure that the timing between taking the interrupt and re-enabling interrupts is greater than the time taken by the VIC interface to complete its handshake. This is 1 cycle at the Vectored Interrupt Controller clock frequency for synchronous operation, or 3 cycles at the Vectored Interrupt Controller clock frequency for asynchronous operation.
4. Finally, extra logic could be added between the processor and the VIC to mask the nIRQ input while IRQADDRV is asserted.

379894: Load Multiple to the PC can be corrupted or result in deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0, Fixed in r0p1.

Description

In ARM1156T2(F)-S, a load multiple to the PC loads the PC in the first transaction of the load multiple. This is done to allow the fetching of the code sequence from the new PC to be performed while the remainder of the words in the load multiple are fetched.

Under rare circumstances a load multiple to the PC will malfunction.

The result is either (a) the loading of the registers other than the PC will be from the wrong location in memory, or (b) a deadlock. This happens because, under the rare circumstances, the remainder of the load multiple is incorrectly treated as being sequential to the load of the PC. This treatment means that they can be fetched from an incorrect cache way, giving incorrect data. Alternatively, if the remainder of the load multiple misses in the cache, the core can deadlock.

Conditions

1. Hit under Miss is enabled (CP15 Control Register FI Bit == 0)
2. An instruction that generates an outstanding memory access. This can be caused by one of:
   - A load which misses in the cache, and whose target register has not been used
   - A PLD operation which has not yet completed
   - A single store outstanding while the Write Buffer is full
3. The following code sequence occurs with the memory access still outstanding.

```
       LDR PC, [Rx]         ; may be a LDM of just the PC.
       …
Target of LDR PC
       ; any number of instructions not including a load or store
       CMP Ra, Rb           ; or any other flag setting operation
       Inst1                ; see note 1
       LDRcond  Rd, [Rf]    ; fails condition codes – see note 2
       LDM Rc, {Rd,…,PC}    ; loads at least 1 register after the PC
                            ; see note 3
```

Note 1: Inst1 is either not present, or is any single cycle instruction other than a load or store

Note 2: This instruction could also be a conditional store or a condition LDM or STM instruction of no more than 2 words

Note 3: This instruction could also be RFE

4. There is a variant of this code sequence where the initial LDR PC, [Rx] is a conditional LDR to the PC or LDM of not more than 3 words, including the PC, which condition code fails.

Implications

This erratum can only occur if Hit under Miss is enabled.

The code sequence needed to generate this erratum can be generated in compiled code. In compiled code, the initial LDR PC, [Rx] is likely to be (but is not limited to) a procedure return.

The failing sequence can happen in both ARM or Thumb code. When in Thumb code, an IT instruction is required to ensure that the Load or Store instruction fails its condition codes.

The required circumstances for this erratum are unusual as there is a need for precisely one outstanding memory operation at the start of this sequence. In general it is unusual for code to have outstanding memory operations when returning from a procedure, but it is a dynamic effect and is correspondingly hard to predict.

When this erratum is triggered the result is either data corruption, or a deadlock.

Workaround

To ensure that this erratum is avoided the design must be configured to disable Hit-Under-Miss. This can be done in one of two ways each with a different impact on the interrupt latency of the processor:

1. Hit-Under-Miss can be disabled by setting the low interrupt latency configuration bit in the CP15 Control Register.
2. To disable Hit-Under-Miss functionality without putting the processor into full low interrupt latency configuration, the undocumented Auxiliary Control Register bit [31] must be set as well as setting the low interrupt latency configuration bit in the CP15 Control Register. This combination of control bits disables the Hit-Under-Miss functionality, but has no effect on the interrupt latency.

Importantly, workaround 2 does not cause multiple word loads and stores to be interrupted part way though. This is not the case with workaround 1. Workaround 2 therefore places no new constraints on the behavior of multiple word loads and stores.

388230: Cache clean when data cache disabled can writeback to the wrong address

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Clean (or Clean and Invalidate) operations performed on the data cache using the address format when the cache is disabled may result in data being written to the wrong address in memory.

Associative cache clean operations are performed in several stages. The first stage checks that the address exists in the cache and if the cache line is clean or dirty. If the address is in the cache, and the cache line is dirty, the second stage of the operation reads the address from the cache TAG RAMs to be used for writing the data back to system memory.

When the cache is disabled, if a load/store operation is performed before the address has been read from the cache for the second operation the dirty cache line may be written back to the wrong address.

Conditions

1. The data cache is disabled.
2. The data cache contains dirty cache lines that have not been invalidated.
3. A Clean Data Cache Line by address (MCR p15, 0, Rx, c7, c10, 1), Clean and Invalidate Data Cache Line by address (MCR p15, 0, Rx, c7, c14, 1), Clean Data Cache Range (MCRR p15, 0, Rx, Ry, c12) or Clean and Invalidate Data Cache Range (MCRR p15, 0, Rx, Ry, c14) operation is performed.
4. A load/store operation is performed before the complete address has been read from the Data Cache. This operation must be the first load/store operation since the cache was disabled

Implications

This erratum can cause memory corruption as data is written to the wrong address in main memory.

Workaround

The erratum may be worked around by executing a Drain Write Buffer Operation (`MCR p15, 0, r9, c7, c10, 4`) immediately after the Cache Clean operation. This prevents any subsequent Load/Store operation from executing until after the clean has completed.

393694: Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Clean and invalidate by range and clean and invalidate entire data cache instructions work by issuing a series of operations to the cache. Each operation will clean and invalidate one cache line. Because these are instructions that can take a long time to execute, it is possible to interrupt them part way through. When the interrupt service routine has completed, it will return to the interrupted clean and invalidate operation which will restart.

This erratum occurs if an interrupt occurs after a specific operation of the instruction has been issued to the cache, but before the instruction has completed. The operation must be either the last operation of the instruction, or at the last address before a 1 kbyte boundary. If the erratum occurs, then the clean part of the operation will be interrupted, but the invalidate part will complete. This will leave the dirty cache line marked as invalid.

Conditions

Condition set 1

1. Bit 21 of the CP15 control register is set to enable low interrupt latency configuration.
2. A clean and invalidate data cache by range (`MCRR p15, 0, Rx, Ry, 14`) or a clean and invalidate entire data cache (`MCR p15, 0, Rx, c7, c14, 0`) instruction is executed.
3. The cache line corresponding to the last operation of the clean and invalidate is dirty.
4. An IRQ or FIQ occurs after the last operation has been issued to the cache, but before the instruction has completed.

Condition set 2

1. Bit 21 of the CP15 control register is set to enable low interrupt latency configuration.
2. A clean and invalidate data cache by range (`MCRR p15, 0, Rx, Ry, 14`) instruction is executed.
3. The cache line corresponding to the last address before a 1k boundary is dirty.
4. An IRQ or FIQ occurs after the last operation in the 1 kbyte region has been issued to the cache, but before the first operation to the next 1 kbyte region has started.

In addition to the above conditions, this erratum requires specific timing between internal signals, and depends on the state of the cache. Therefore, any code that replicates the given conditions may not stimulate this erratum.

Implications

If this erratum occurs, the cache line will still be dirty, but it will be marked as invalid. Therefore when the interrupt service routine returns, and the clean and invalidate instruction is restarted, it will not clean the dirty line. The data that was in the cache line will never get written to external memory.

Workaround

There are several possible workarounds to ensure that this erratum is avoided:

1. Low interrupt latency configuration can be disabled by clearing bit 21 in the CP15 Control Register.
2. Low interrupt latency configuration can be disabled without enabling Hit-Under-Miss functionality, by using the undocumented Auxiliary Control Register bit 31 which must be set as well as setting bit 21 in the CP15 Control Register.
3. Interrupts can be disabled for the duration of the clean and invalidate instruction by setting the I and F bits in the CPSR. This workaround is only possible when the FIQISNMI configuration pin is low.
4. A clean and invalidate by range instruction can be replaced by a sequence of clean and invalidate single line instructions.
5. A clean and invalidate all instruction can be replaced by a clean all instruction followed by an invalidate all instruction.

Workaround 3 will have a large impact on interrupt latency whilst the clean and invalidate instruction is in progress, but will not affect latency at other times.

Workarounds 1 and 2 will affect interrupt latency for all instructions, but will still allow clean and invalidate instructions to be interrupted without triggering this erratum.

For workarounds 4 and 5, you may need to read the cache dirty status register to ensure that interrupt service routines have not dirtied the cache.

393696: DTCM stalls may cause data corruption

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The level one write buffer contains data for stores that have been executed, but haven't yet been written to cache or TCM. If a load instruction is executed to the same address as an entry in the write buffer, it will load the data from the write buffer rather than from the cache. If two stores are executed to addresses in the same double word, and are both contained in the write buffer, then when a load is executed the write buffer will merge the data from the two entries to provide the correct data.

This erratum occurs when the first write in the write buffer is presented to the TCM, but the TCM is stalled because of a load or store in the previous cycle has resulted in the assertion of the nDTCDATARDY. This can result in data corruption.

Conditions

There are two sets of conditions that can cause this erratum:

Condition set 1

1. A store instruction is executed to an address in DTCM, and gets placed in the write buffer.
2. This is either a store multiple, or there are other stores following it, so that the write buffer becomes full.
3. A load or store is executed to an address in DTCM. This must be either:
    - a load which causes a TCM stall, signalled by the assertion of nDTCDATARDY, OR
    - a store which causes the head of the write buffer to drain to TCM, and the store at the head of the write buffer causes a stall, signalled by the assertion of nDTCDATARDY
4. A store instruction is executed to an address within the same double word as the first store which is now at the head of the write buffer.
5. A load instruction is executed to an address in the same double word as the two entries in the write buffer.

Condition set 2

1. A store instruction is executed to an address in DTCM or cache, and gets placed in the write buffer.
2. A second store instruction is executed to an address in DTCM or cache, and gets placed in the write buffer. Alternatively, a clean or clean and invalidate cache maintenance instruction is executed and gets placed in the write buffer.
3. The first store is drained from the write buffer the cycle after a previous load or store caused a DTCM stall signalled by the assertion of nDTCDATARDY.

For condition set 1, the entries for the two stores to the same double word may have their byte strobes merged incorrectly. If a load reads from this incorrectly merged data while it is still in the write buffer, then it will receive corrupted data. The writes will still happen to the TCM correctly.

For condition set 2, the first entry in the write buffer can get the way associated with it corrupted. This means that it can be written to the wrong way in the cache, or it could be written to the TCM when it should have been written to the cache, or it could be written to the cache when it should have been written to the TCM.

Implications

The implications of the erratum is that the assertion of nDTCDATARDY can lead to data corruption if the DTCM contains writeable data.

If nDTCDATARDY is not being used, or is only asserted as a prelude to an unrecoverable error (to be signalled on DTCDATAERROR[7:0]) then the erratum has no effect. The erratum also has no effect if the DTCM is being used in a read-only manner.

Workaround

No workaround available other than not use the nDTCDATARDY stalling mechanism.

393697: Data cache dirty RAM parity error on a write through region of memory can write back stale data

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Data cache lines that contain memory marked as write through cannot contain dirty data. However, if there is a parity error on the dirty RAM then they appear as if they were dirty.

In the absence of errors, a store to write through memory that hits in the cache will write the data to external memory, and also to the level one write buffer to update the cache. Later, when the write buffer is drained, the data will be written to the cache.

This erratum occurs if there is a store to a write through region of memory that hits in the cache. The data is written to external memory. Whilst the data is still in the write buffer, the cache line that it will be written to is evicted by a load causing a linefill. If there is a dirty error on the line being evicted, it will be written back to external memory, overwriting the data written by the store.

Conditions

1. The PE bit in the CP15 auxiliary control register is set to enable parity errors.
2. A store instruction is executed to a region of memory marked as write through.
3. The store hits in the cache.
4. After the store has been written to external memory, but before it has been written to the cache, the line it was going to be written to is evicted.
5. The evicted line has a parity error on the dirty RAM.

Implications

A parity error is a rare event. An eviction happening to a cache line that is going to be written to by the write buffer is possible with any size of data cache, however it is much more likely to happen with 1k or 2k cache as they have fewer ways than other cache sizes.

If this erratum occurs, external memory will be corrupted with no indication to the software. No data abort will occur because dirty errors do not cause aborts.

Workaround

No workaround available.

393698: VFP or GCP instruction that stalls may not trace its data with the ETM

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

VFP or GCP instructions can be stalled by the coprocessor, if it is busy or interlocked. If the ETM is in use then the data transferred to or from the coprocessor can be traced once the coprocessor has ended the stall. If the coprocessor instruction is followed immediately by a load or store instruction that fails its condition code, then the data for the coprocessor instruction may be missing from the ETM trace.

Conditions

1. A flag setting instruction is executed.
2. An MCR, MRC, LDC, or STC instruction is executed to the VFP or a GCP.
3. The coprocessor instruction is stalled by the coprocessor for at least one cycle.
4. A load or store instruction is executed which fails its condition codes.

Implications

The behavior of the processor is not affected. The data associated with the coprocessor instruction will not appear in the ETM trace, and any comparators set to trigger on it will not trigger. Subsequent data items may get associated with the wrong instructions in the ETM trace, and trace synchronisation may be lost.

Workaround

If the ETM is not in use then no workaround is needed. If the instruction after the coprocessor instruction is not a load or store then the erratum will not occur.

395146: VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction

Status

Affects:	product(s) ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

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 ARM1156T2F-S 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 IEE754 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.

397735: Write back data cache entry evicted by write through entry causes data corruption

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The data cache dirty bits indicate that a cache entry is in a write back region of memory and that the cache entry has been updated since being allocated to the cache. The dirty bits for a cache entry are cleared at the start of a cache linefill. If there is a write in the level 1 memory write buffer for a cache entry that is being replaced by a write through entry, it is possible for the dirty bits for the new, write through, entry to be incorrectly set and mark the entry as dirty.

A subsequent write to the write through entry will update the external memory and enter the L1 write buffer. If the cache entry is evicted from the cache before the write has been pushed from the L1 write buffer the stale data resident in the cache will be written back to external memory and the write in the L1 buffer will be lost.

Conditions

1. The memory map has both write through and write back memory regions.
2. A write back entry is present in the cache.
3. A write to the cache entry is at the head of the level 1 write buffer.
4. A read of a write through address results in a cache miss and the eviction of the write back cache entry.
5. A write to an unrelated cache entry pushes the head entry from the write buffer after the linefill for the write through address has started.

It should be noted that the write back cache entry correctly writes back to external memory and that the write through data is not corrupted by pushing the entry from the L1 write buffer.

Implications

This erratum can lead to the corruption of data memory as a result of the write back of stale data on a cache eviction.

Workaround

The erratum may be avoided by marking all cacheable memory as one of write through or write back. This requires the MPU region attributes to be modified by software appropriately.

397751: Data cache clean operations may not abort on tag or data RAM parity errors

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Data cache clean instructions must read the tag and data RAMs to determine the address and data to write back to external memory. If a parity error is detected in these RAMs, then the potentially incorrect data or address should not be written out to external memory. A data abort should be generated to inform the software that an error was detected.

This erratum can in some situations cause a parity error to be ignored, and the incorrect data to be written to external memory.

Conditions

There are two sets of conditions that can cause this erratum.

Condition set 1:

1. The memory map used includes areas of memory marked as write back cacheable.
2. A clean entire data cache (`MCR p15, 0, Rx, c7, c10, 0`), clean and invalidate entire data cache (`MCR p15, 0, Rx, c7, c14, 0`), clean range (`MCRR p15, 0, Rx, Ry, 12`), and clean and invalidate range (`MCRR p15, 0, Rx, Ry, 14`) instruction is executed.
3. A parity error is present on the data RAM of an address being cleaned. The error is not in the first doubleword of the cache line.
4. The cache line being cleaned is dirty.
5. The next cache line to be accessed by the instruction is not valid.

Condition set 2:

1. The memory map used includes areas of memory marked as write back cacheable.
2. A clean entire data cache (`MCR p15, 0, Rx, c7, c10, 0`) instruction is executed.
3. A parity error is present on either the tag RAM or the data RAM of an address being cleaned.
4. The cache line being cleaned is dirty.
5. The next cache line to be accessed by the instruction is not valid.

In addition to the above conditions, this erratum requires specific timing between internal signals, and depends on the state of the cache. Therefore, any code that replicates the given conditions may not stimulate this erratum.

Implications

If a tag or data parity error occurs for a cache line that is dirty with the conditions for this erratum, then it is possible for external memory to be corrupted. Software will be unaware of this as no abort will have been generated.

Workaround

No workaround available.

398952: Tag or valid parity error on a load multiple to the PC can deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1, Fixed in r0p2.

Description

A multiple word load may cross one or more cache line boundaries. For each boundary crossed, the processor must read the tag and valid RAMs to determine if the data is in the cache, and which way it is in. If the load includes the PC in the register list, an optimisation in the ARM1156 causes the load of the PC to be executed first, followed by the load of the remaining registers in the original specified order. This means that it is possible for the tag and valid RAMs to be read twice for the cache line containing the PC.

If a parity error is detected in the tag or valid RAMs the first time they are accessed when the PC is read, then a data abort will correctly be taken. However, if there is no parity error when they are accessed for the PC, but there is a parity error detected in the same RAM when it is read a second time for the same instruction, then this erratum can cause the processor to deadlock.

Conditions

1. The processor is not in low interrupt latency configuration.
2. Parity checking is enabled in the auxiliary control register.
3. An LDM or POP of at least three registers (including the PC) is executed.
4. Data alignment is such that the PC and at least one other word are situated in one cache line, and at least one other word is situated in a different cache line (or is not present in the cache).
5. No parity error is detected when the PC is read from the cache.
6. A parity error is detected in the tag or valid RAMs when they are read for a second time for a later part of the same instruction.

Implications

A parity error is a rare event. The time between the start of the load multiple and the end of the load multiple instruction is likely to be quite short. The parity error must occur on the cache line being accessed, in the short period while the instruction is executing. This combination is going to have an extremely low probability of occurring.

Workaround

No workaround available.

414935: Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1,r0p2, Fixed in r0p3.

Description

The ARM1156 processor implements a four-way set associative cache. For the purpose of this erratum, a cache line group is defined as eight cache lines whose set differs only in the bottom bit. For example, the following eight cache lines form a cache line group:

- Set 0xbe, Way 0
- Set 0xbe, Way 1
- Set 0xbe, Way 2
- Set 0xbe, Way 3
- Set 0xbf, Way 0
- Set 0xbf, Way 1
- Set 0xbf, Way 2
- Set 0xbf, Way 3

If an instruction cache linefill completes at the same time as an Invalidate Entire Instruction Cache operation, the cache line being filled and the other cache lines within the same cache line group might not be invalidated.

Conditions

The following conditions must all be met:

1. An instruction cache linefill occurs. This can occur if:
   - The processor executes an instruction which is not in the instruction cache. The processor will fetch the instruction to be executed first, and then fetch the other instructions in the same cache line.
   - The processor prefetches instructions after the instruction being executed which are not in the instruction cache, ready for future execution.
2. An Invalidate Entire Instruction Cache operation is performed.
3. The Invalidate Entire Instruction Cache operation completes on the same cycle that the linefill completes.

Implications

Under rare circumstances, systems which rely on Invalidate Entire Instruction Cache operations to permit reuse of the same address space for different instruction sequences might fail. Systems which rely on these operations include those which use self-modifying code or implement paging. When replacing one set of instructions with another set of instructions in memory, some of the first set of instructions might remain in the cache.

If the code which performs the invalidate operation is executed in the same way each time, it is likely that this erratum will not occur, because of the combination of timings required to hit it. However, factors such as interrupts and memory system stalls can change the timing of the invalidate operation. Therefore it must be assumed that all systems which use Invalidate Entire Instruction Cache operations are potentially affected by this erratum.

Workaround

Workaround 1

Replace the Invalidate Entire Instruction Cache operation with:

```
MOV Rtmp1, #0
MRS Rtmp2, cpsr
CPSID ifa                     ; disable interrupts
MCR p15, 0, Rtmp1, c7, c5, 0   ; Invalidate Entire Instruction Cache
MCR p15, 0, Rtmp1, c7, c5, 0   ; Invalidate Entire Instruction Cache
MCR p15, 0, Rtmp1, c7, c5, 0   ; Invalidate Entire Instruction Cache
MCR p15, 0, Rtmp1, c7, c5, 0   ; Invalidate Entire Instruction Cache
MSR cpsr_cx, Rtmp2            ; reenable interrupts
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

This workaround can only be guaranteed to work in all cases if the NMFI bit in the System Control Register is clear. If this bit is set then FIQs cannot be disabled. Note that this bit is a read-only bit indicating the configuration of the processor.

This workaround has been designed to be applied to ARM1136, ARM1156 and ARM1176 family processors. ARM11 MPCore is not affected by this erratum. ARM Ltd recommends that you do not apply this workaround to code sequences which might be run on ARM11 MPCore because the Invalidate Entire Instruction Cache operation on ARM11 MPCore takes many clock cycles, and the workaround might therefore impact performance.

The performance impact of this workaround when run on ARM1136, ARM1156 and ARM1176 family processors is low. The EEMBC suites have been run and have shown no statistically significant difference in performance. Individual benchmark results range from 0.16% worse to 0.18% better, with the vast majority varying by no more than 0.03%.

This workaround is explained as follows:

- Interrupts are disabled to prevent instruction cache linefills caused by servicing an interrupt.
- The invalidate operation is performed four times. It is not possible for this erratum to occur four times in a row, because this would require four instruction cache linefills:

    - A linefill might be outstanding upon entry to the workaround sequence, and complete at the same time as the first invalidate operation, causing the erratum to occur.
    - A second linefill might be started as a result of the second invalidate operation, and in extreme circumstances might complete at the same time as the second invalidate operation, causing the erratum to occur for a second time.
    - A third linefill might be started as a result of prefetching past the cache line containing the second invalidate operation, and in extreme circumstances might complete at the same time as the third invalidate operation, causing the erratum to occur for a third time.
    - It is not possible for a fourth linefill to be started, due to the number of prefetch buffers implemented.
- A sequence of NOPs is used to ensure that, while executing the invalidate operations, the processor will not encounter a predicted taken branch which might cause an additional instruction cache linefill. The number of NOPs is calculated based upon the requirements of ARM1136, ARM1156 and ARM1176 family processors.

Workaround 2

Replace the Invalidate Entire Instruction Cache operation with:

```
MOV Rtmp1, #0
MRS Rtmp2, cpsr
CPSID ifa                      ; disable interrupts
MRC p15, 0, Rtmp3, c1, c0, 0   ; read control register
BIC Rtmp4, Rtmp3, #0x1000
MCR p15, 0, Rtmp4, c1, c0, 0   ; disable instruction cache
MCR p15, 0, Rtmp1, c7, c5, 4   ; flush prefetch buffer
MCR p15, 0, Rtmp1, c7, c5, 0   ; invalidate Entire Instruction Cache
MCR p15, 0, Rtmp3, c1, c0, 0   ; reenable instruction cache
MSR cpsr_cx, Rtmp2             ; reenable interrupts
```

This workaround disables the instruction cache while the invalidate is in progress. If the F bit cannot be cleared because the NMFI bit in the System Control Register is set, then there is a risk that the FIQ handler will be executed with the instruction cache disabled. Therefore it is recommended that this workaround is only used in systems where the FIQ handler resides in the instruction TCM.

If NMFI bit is clear then it is recommended that you implement workaround 1 instead.

Workaround 3

Ensure that the Invalidate Entire Instruction Cache operation is executed from non-cacheable memory, and that it is not within 12 instructions of a BTAC predictable branch to cacheable memory.

This workaround works because the execution of instruction from non-cacheable memory causes a new instruction fetch, which cannot occur until any outstanding linefills have completed. It is not sufficient to execute the invalidate operation from the instruction TCM because this does not cause a new instruction fetch.

720619: Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1,r0p2,r0p3,r0p4, Open.

Description

Clean by MVA operations to the data cache can be performed by writing to CP15 register 7. These operations perform a cache lookup using the supplied MVA. If a cache line is hit, any dirty data in the cache line is written back to external memory and the cache line is marked as clean. If only half of the cache line is marked as dirty then only the data in that half will be written back.

Hit-Under-Miss allows program execution to continue after there has been a data cache miss. Hit-Under-Miss is enabled by default after reset and can only be disabled by setting the FI bit (bit [21]) in the CP15 Control Register.

The Hit-Under-Miss functionality makes it possible for a store instruction located after a Clean Data Cache Line by MVA instruction to write to the cache before the Clean completes. If the cache line is marked as half-dirty before the instructions are executed then the Clean will have to write the data in the dirty half of the cache line back to external memory. If both instructions access the same cache line and the store causes the entire cache line to be marked as dirty, it is possible for the Clean to still only write back half of the cache line to external memory. The data from the store will then be marked as clean in the cache line even though it may not be synchronized with the corresponding location in external memory, possibly resulting in data corruption if the cache line is later evicted or invalidated.

Conditions

The following conditions must all be met:

1. Hit-Under-Miss is enabled.
2. A cache line exists that is valid and partially dirty.
3. A Clean Data Cache Line by MVA (MCR p15, 0, Rx, c7, c10, 1) instruction is executed to clean the cache line.
4. A store instruction to the part of the cache line that was not dirty is executed at some point after the Clean instruction.

In addition to the above conditions, this erratum requires specific timing between internal signals and on the external AXI buses. Therefore any code that replicates the given conditions might not stimulate this erratum.

Implications

The store data might be lost under the conditions described above, resulting in data corruption.

Clean and Invalidate Data Cache Line by MVA operations are not affected by this erratum.

Data cache evictions are not affected by this erratum.

Clean Data Cache Line by MVA operations are commonly followed by a DSB to ensure visibility of the cleaned data. This prevents the erratum from occurring.

Workaround

Two workarounds are possible for this erratum:

1. Execute a DMB (`MCR p15, 0, Rx, c7, c10, 5`) or DSB (`MCR p15, 0, Rx, c7, c10, 4`) between the Clean Data Cache Line by MVA instruction and the next store instruction to the cache line that was cleaned.
2. Disable the Hit-Under-Miss functionality by setting the FI bit in the CP15 Control Register. To avoid putting the processor into full low interrupt latency mode, the FIO bit (bit [31]) of the Auxiliary Control Register must also be set. This combination of control bits disables the Hit-Under-Miss functionality but has no effect on the interrupt latency.

The first workaround will result in a minor decrease in the performance of Clean instructions. The second workaround will decrease the instruction throughput when a data cache miss occurs, meaning that it will have a negligible effect on programs with effective data cache usage.

720627: Invalidate Instruction Cache operations can fail

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 2, Present in: r0p0,r0p1,r0p2,r0p3,r0p4, Open.

Description

Under very rare conditions, Invalidate Instruction Cache entry operations can fail to invalidate a cache.

The following CP15 operations are affected:

1. Invalidate Instruction Cache line by MVA (`MCR p15, 0, <rd>, c7, c5, 1`)
2. Invalidate Instruction Cache range (`MCRR p15, 0, <end_addr>, <start_addr>, c5`)
3. Invalidate Instruction Cache line by set/way (`MCR p15, 0, <rd>, c7, c5, 2`)

Conditions

1. The cache line to be invalidated has been fetched from the L2 memory but not committed to the instruction cache
2. An Invalidate Instruction Cache line or Invalidate Instruction cache range operation is executed

It should be noted that particular timings of events are required in addition to the above conditions for the erratum to be triggered. As a result, not all sequences that meet these conditions will trigger the erratum.

Implications

Associative instruction cache invalidate operations might not correctly invalidate the instruction cache, which can lead to incorrect program execution.

The cache maintenance operations are expected to be contained within HAL code and require the processor to be in a privilege mode to be executed. This makes analysing code and applying workarounds more straightforward.

Workaround

The following workarounds can be applied for this erratum.

Workaround 1

Execute the invalidate instruction twice. For example:

```
MCR p15, 0, R0, c7, c5, 1
```

becomes:

```
MCR p15, 0, R0, c7, c5, 1
MCR p15, 0, R0, c7, c5, 1
```

Workaround 2

Use the Invalidate Entire Instruction Cache instruction instead. The workaround for erratum 414935 must be used if that erratum is applicable to the core revision being used.

Errata - Category 3

333141: VFP can take an inexact exception on non-CDP VFP instructions

Status

Affects:	product(s) ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r0p4, Open.

Description

If the VFP FPSCR register IXE bit is set, all VFP CDP instructions are redirected 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 redirected incorrectly.

Conditions

1. The VFP is enabled
2. The IXE bit is set to enable inexact exceptions to be trapped
3. A VFP CDP operation (or equivalent bit pattern) appears in a branch or exception shadow

Implications

The use of the VFP IXE bit to trap all inexact exceptions results in all VFP arithmetic operations being handled by software support code. Therefore applications are unlikely to set the IXE bit.

Where the IXE bit is set, this results in some VFP load and store operations being redirected 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.

374844: STREX to ITCM that fails its condition codes immediately followed by a STM to SO/DEVNS memory may cause a deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

If a STREX instruction is executed to an address that is in ITCM, and it fails its condition code, then an immediately following STM to strongly ordered or device memory may cause a deadlock.

Note that a STREX to ITCM is documented as having UNPREDICTABLE behavior, therefore correctly written code should not be doing a STREX to ITCM.

Conditions

1. A flag setting instruction is executed.
2. A conditional STREX instruction is executed, which fails its condition code.
3. The STREX must be either the first instruction after the flag setting instruction, or there can be one single cycle instruction between the flag setting instruction and the STREX.
4. A store multiple instruction is executed to strongly ordered memory.
5. The store multiple must be at least 6 words long.
6. The sixth or seventh word in the store multiple must be aligned to a 32 byte boundary.

In addition to these conditions, specific timing of internal signals is required to stimulate this erratum.

The processor will deadlock on word five or word six of the store multiple to strongly ordered memory depending on the alignment of the data.

Implications

If this erratum occurs the Processor will stall and not process this or any further instructions. However this erratum will not occur when executing correctly written code.

Workaround

For correctly written code, no workaround is needed.

376081: ITCM error caused by data access is not counted by performance monitor

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The System Performance Monitor is used to monitor events occurring in ARM1156T2F-S and ARM1156T2-S. Event number 0x34 counts parity errors received from the ITCM. However, only errors received due to instruction requests are counted, errors received due to data loads are not counted. This means that parity errors occurring in data regions of the ITCM cannot be counted. However they will still cause data aborts.

Conditions

1. One of the System Performance Counters is programmed to count ITCM parity errors.
2. A load from the ITCM (e.g. literal pool access) is performed which generates an error.

Implications

The System Performance Monitor will not count all ITCM errors. Data load and instruction fetch aborts occur as expected.

Workaround

Although the performance monitor does not count the error, the data abort handler is called as normal. Therefore a counter can be maintained in software. The abort handler code can increment this software counter each time the abort handler is called due to an error. The abort handler can determine if an abort was due to an ITCM access by examining the address in the Fault Address Register. It can determine if the abort was due to a parity error by examining the Data Fault Status Register.

376232: DTCM error counted multiple times by performance monitor whilst DTCM stalled.

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The Performance Monitor counters can be used to count the number of DTCM parity errors that occur. The TCM signals an error by asserting the DTCDATATERROR input to the ARM1156 processor. The TCM can also cause a stall by asserting the nDTCDATARDY input to the ARM1156 processor. Partners may implement the TCM such that nDTCDATARDY is asserted whenever DTCDATAERROR is asserted, while the TCM attempts to correct the error. The processor will only cause a data abort to be taken if DTCDATAERROR is still asserted when nDTCDATARDY has been deasserted. However, the Performance Monitor will incorrectly increment for every cycle that both DTCDATAERROR and nDTCDATARDY are asserted.

Conditions

1. At least one of the performance monitor counters is programmed to count DTCM errors.
2. A read access to the DTCM causes the nDTCDATARDY and DTCDATAERROR inputs to be asserted.

Implications

This erratum only occurs if a partner implements the TCM such that DTCDATAERROR and nDTCDATARDY may be asserted at the same time. If it occurs, the Performance Monitor counter may show a higher count than the actual number of errors that have occurred.

Workaround

The exact workaround will depend on how the partner has implemented the TCM. For example, if the implementation always stalls for a fixed number of cycles when an error is encountered then this knowledge can be used to adjust the result.

Alternatively, a counter can be maintained in software. The data abort handler will be called when the error is reported to the processor. Therefore the abort handler code can increment this software counter each time the abort handler is called due to an error. The abort handler can determine if an abort was due to a DTCM access by examining the address in the Fault Address Register. It can determine if the abort was due to a parity error by examining the Data Fault Status Register.

378833: Performance monitor does not count instruction cache errors when set to reload on error

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The System Performance Monitor is used to monitor events occurring in ARM1156T2F-S and ARM1156T2-S. Event number 0x31 counts parity errors received from the instruction cache data rams.

The IR Bit (bit 6 of the CP15 Auxiliary Control Register) is used to control the behavior when a parity error is detected in one of the instruction cache data rams. When it is set parity errors in the instruction cache data rams cause an automatic refetch of the relevant cache line.

While the IR Bit is set parity errors in the instruction cache data rams are not counted by the System Performance Monitor.

Conditions

1. At least one of the performance monitor counters is programmed to count instruction cache data errors.
2. The IR bit (bit 6 of the CP15 Auxiliary Control Register) is set to 1 (refresh on instruction cache error).
3. The instruction cache is enabled.
4. An instruction fetch is made which hits in the cache.
5. The RAM entry for the instruction fetch has a parity error.

Implications

When the IR Bit is set the System Performance Monitor will not count all parity errors in the instruction cache data rams. Instruction cache refetch on a parity error will occur as expected.

Workaround

To work around this problem the IR Bit should be set to 0 and the prefetch abort handler should invalidate the cache line which caused the abort before returning execution to the location of the prefetch abort. In this case instruction cache data ram parity errors will be counted as expected because the IR Bit is not set.

380083: Enabling the instruction cache during an external fetch may result in duplicate cache entries

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The instruction cache can be enabled or disabled at any time by writing to bit 12 of the CP15 control register. If the instruction cache is enabled while a fetch from level two memory is in operation there is a possibility that data will be duplicated in the cache rams.

Conditions

1. The instruction cache is initially disabled.
2. The MPU is enabled.
3. The instruction cache is then enabled by writing to the CP15 control register.
4. At the time the CP15 write is performed the PFU starts a new instruction fetch from level two memory.
5. The MPU region for this fetch is cacheable.
6. The data for this fetch is already in the instruction cache.

In addition to the above conditions, specific timing between internal signals is also required to stimulate this erratum. Therefore, any code that replicates the given conditions may not stimulate this erratum.

This will result in multiple ways in the instruction cache containing the same tag and data values. The data and tag values will still be correct but duplicated.

Implications

Duplicate data in the instruction cache will not result in corrupted data so long as ARM guidelines on the use of caches are followed. Duplicate data entries will only be apparent to the programmer if direct access is made to the instruction cache rams using the CP15 instruction cache debug register. A compiler would not normally generate code to enable or disable the instruction cache, nor would it use direct access to the instruction cache rams, so would not stimulate this erratum.

Workaround

This erratum can be worked around by ensuring that no requests to cacheable MPU regions are occurring at the time that the instruction cache is enabled. This may be done by executing the following sequence of instructions from either an area of memory marked as non-cacheable or from the ITCM. The NOPs ensure that following code cannot run into or jump into a cacheable region.

```
MCR p15, 0, R0, c7, c5, 4 ; Flush prefetch buffer
MRC p15, 0, R0, c1, c0, 0 ; Read control register
ORR R0, R0, #0x1000       ; Set bit 12
MCR p15, 0, R0, c1, c0, 0 ; Write control register
NOP
```

```
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

Alternatively, any code that uses the instruction cache debug register can be updated to work correctly if the instruction cache contains duplicate entries.

If no code makes use of the instruction cache debug register, then no workaround is needed.

389801: Cache clean after reprogramming the MPU may cause subsequent instructions to fail

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Performing a cache maintenance operation (clean or clean and invalidate) to an address that has had its memory attributes changed to Strongly Ordered may result in subsequent instructions failing.

If an address initially has normal, cacheable write-back attributes and the data cache is enabled it will be allocated into the data cache on a read. If the memory attributes for the address are then changed to Strongly Ordered memory by reprogramming the MPU, a cache maintenance operation to the address will cause the premature retirement of load/store operations.

The failure may manifest as instructions retiring too early causing data aborts to be missed, or in the worst case a deadlock situation may occur.

Conditions

1. An address region is initially marked as non-shared, write back cacheable.
2. An address within this region is allocated to the data cache.
3. The attributes for the memory region are changed to Strongly Ordered Memory.
4. A cache clean or clean and invalidate by address operation is performed.

Implications

This erratum may result in instructions being retired too early with subsequent loss of abort information. In extreme cases it may cause a deadlock.

The erratum will not occur in correctly constructed code. Code to reprogram the MPU or perform cache maintenance operations will not normally be part of user code.

Changing the memory attributes in the way required to provoke this erratum is architecturally unpredictable and so should not exist in current code.

Workaround

The erratum can be avoided by cleaning (and/or invalidating) affected addresses before changing the memory attributes in the MPU to strongly ordered.

391562: Vector catch on IRQ may not trigger when the VIC is enabled

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

For the ARM1156T2F-S and ARM1156T2-S processors the location in memory of the IRQ exception vector can be determined directly from the value presented on the Vectored Interrupt Controller (VIC) port. This vector interrupt behavior is explicitly enabled when the VE bit in the CP15 control register is set.

The ARM1156T2F-S and ARM1156T2-S processors support exception vector catching, including IRQ exceptions. These are enabled using the Vector Catch Register (VCR) in CP14, c7. These should occur whether or not a Vectored Interrupt Controller is in use.

If a FIQ exception occurs after the IRQ exception but before the first instruction of the IRQ handler is executed, the vector catch on the IRQ exception should be postponed until the FIQ handler has finished. However, this erratum prevents the IRQ vector catch from occurring as expected when a Vectored Interrupt Controller is in use.

Conditions

1. The Vectored Interrupt Controller is in use and enabled using VE bit in the CP15 control register.
2. A Vector catch has been set for IRQ exceptions using the CP14 c7 Vector Catch Register.
3. An IRQ exception occurs.
4. A FIQ exception occurs after the IRQ exception but before the first instruction of the IRQ handler is executed.

In this case the IRQ exception vector catch does not occur when the IRQ handler is executed after the FIQ handler has returned.

Implications

When a Vectored Interrupt Controller is in use some IRQ vector catches may be missed. This will only affect debugging operations using RealView or a similar tool.

If a Vectored Interrupt Controller is not in use IRQ vector catches will occur as expected.

Only IRQ vector catches are missed, the IRQ vector is executed as expected.

Workaround

To workaround this erratum, the FIQ vector catch should be enabled whenever the IRQ vector catch is enabled. When a FIQ exception occurs, it will cause a vector catch, and the SPSR and R14 values can be examined. If they match IRQ mode and an address that is known to be the start of an IRQ handler routine then an IRQ has occurred. Alternatively, if there are only a small number of IRQ handler routines then a breakpoint can be set on the first instruction of each IRQ handler.

391890: Interrupted folded branch may be missed from ETM trace

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r0p4, Open.

Description

If an interrupt request occurs during an operation that cannot be interrupted, then the interrupt will normally be taken on the next instruction. However, if the next instruction is a folded branch, IT, or NOP, then in some circumstances the interrupt may be taken on the instruction following the folded instruction. This can occur when the branch prediction is known to be correct, and avoids having to reexecute the folded instruction when the interrupt handler returns. This gives an improvement in performance.

If this erratum occurs, the folded instruction will not be present in the ETM trace.

Conditions

1. Bit 21 of the CP15 control register is set to enable low interrupt latency configuration.
2. An MCR, MRC, LDC or STC instruction to CP14 or CP15 is executed.
3. This is followed by a branch, IT or NOP instruction.
4. The branch, IT or NOP is folded onto the next instruction.
5. If the folded instruction is a branch, it is correctly predicted.
6. An IRQ or FIQ request occurs whilst the CP14/15 operation is in progress.
7. The ETM is in use to trace the instruction stream.

Implications

This erratum does not affect the behaviour of the processor, it only affects what is reported to the ETM.

If the erratum occurs, the branch instruction will be missing from the ETM trace. If any ETM address comparators are programmed for the address of the branch instruction, they will not match. Because the missing instruction is caused by an interrupt, it will always be followed by an exception packet in the trace stream. Therefore, trace decompressors will not lose instruction synchronisation.

Workaround

There is no need to work around this erratum.

393338: A prefetch after an aborted prefetch may be ignored

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Prefetch operations to the instruction cache can be performed by writing to CP15 register 7. If the required address is not in the instruction cache a request will be made on the AXI interface. If this request receives an abort (either SLVERR or DECERR) the linefill will not write to the cache but this is not visible to the user.

This erratum concerns a prefetch operation occurring immediately after an aborting prefetch operation. If no AXI abort is received the second prefetch should cause a linefill as expected. However, in some cases the second prefetch is ignored.

Conditions

1. A prefetch instruction (`MCR p15, 0, Rx, c7, c13`) is executed.
2. The prefetch receives an AXI abort, and does not alter the cache.
3. A second prefetch instruction is executed after the first has started but before the corresponding AXI burst has finished (the AXI read burst is not terminated by a read error).

The second condition requires that the AXI burst of four lasts more than eight cycles. This could be due to a clock ratio of more than 2:1 or more than one wait state per read.

Implications

The failure of the prefetch operation can only affect performance due to a cache line not being loaded into the instruction cache when expected. In most cases the failure of a prefetch operation will not cause significant reduction in performance because the cache line will be loaded when the code is executed.

If one or more cache ways are locked down after the prefetch operations have finished the required address may not be in the locked down cache way and so will be loaded into one of the non-locked down cache ways when it is needed.

Vital prefetch operations should not normally be made to AXI memory which is likely to respond with an abort (SLVERR or DECERR).

AXI aborts due to requests other than prefetch operations (for example due to normal fetch operation) will not cause this erratum.

Workaround

If it is vital that every prefetch operation successfully loads data into the instruction cache then the instruction cache rams can be checked after the prefetch operations have been executed. This can be done using Instruction cache debug operations as described in the Technical Reference Manual. The relevant instructions are:

```
MCR p15, 3, <Rd>, c15, c2, 1 ;Instruction cache Tag RAM read operation
```

where <Rd> contains the required way and set numbers

```
MRC p15, 3, <Rd>, c15, c0, 1 ;Read Instruction Cache Debug Register
```

Which will load into <Rd> the tag address and valid bit which can be checked against the address of the prefetch operation. Note that a prefetch operation can load into any of the four cache ways unless some of those ways have been locked down.

Alternatively the prefetch operations can be performed while executing from non-cacheable memory and a Flush Prefetch Buffer instruction executed after each prefetch operation:

```
MCR p15, 0, <Rd>, c7, c5, 4
```

This will cause a normal fetch to occur between each prefetch operation.

393703: Conditional load or store instructions executed after reset may deadlock

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The N, Z, C, and V flags in the CPSR have UNPREDICTABLE values after the ARM1156T2(F)-S is reset. Therefore, executing a conditional instruction before an instruction has written to the flags results in UNPREDICTABLE behaviour.

However, if a conditional load, store, or coprocessor transfer instruction is executed before any instruction has written to the flags then subsequent instructions may fail or cause a deadlock.

Conditions

1. The processor is reset.
2. No flags setting instructions are executed.
3. A conditional load, store, MCR, MRC, LDC, STC instruction is executed. The condition code must be GE, LT, GT, or LE.

In addition to instructions that explicitly update the flags, other instructions such as those that update the GE flags, or loads to the PC, may refresh the value of the flags which is sufficient to avoid this errata.

Implications

This erratum may result in following instructions being retired too early with subsequent loss of abort information. In extreme cases it may cause a deadlock.

The erratum will not occur in correctly constructed code.

Workaround

No workaround is needed for correctly written code.

394384: Invalidate data cache line by address does not cause an abort on a valid error

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

Data cache invalidate by address instructions must do a cache lookup to determine which cache line to invalidate. If there is a parity error detected on the valid RAM during this lookup, it should cause a data abort. This erratum causes no abort to be generated.

Conditions

1. An invalidate data cache line by address instruction (`MCR p15, 0, Rx, c7, c6, 1`) is executed.
2. The cache line associated with that address has a parity error on the valid RAM.

Implications

An abort will never be generated from the error, therefore software will not be aware that an error occurred.

As the cache line was being invalidated, any data in the cache line was not required by the software. Therefore software would not normally be concerned as to whether there was an error or not.

Workaround

No workaround available.

394634: Return stack events not counted correctly by performance monitor

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The System Performance Monitor is used to monitor events occurring in ARM1156T2F-S and ARM1156T2-S. Event number 0x36 counts procedure return addresses popped off the return stack, and event 0x37 counts procedure return addresses popped off the return stack that have been incorrectly predicted. This erratum can in some cases cause the counted events to be lower than they should be. Mispredicts caused by `LDR pc`, `LDM r13,{...,pc}`, and `POP` instructions are counted correctly, however mispredicts caused by `MOV pc, r14` and `BX r14` instructions are not counted by the performance monitor. If the mispredict happens before the target instruction of the prediction has been fetched, then it is possible for the mispredict event to be counted, but the return stack pop event is not counted.

Conditions

There are two sets of conditions that can cause incorrect event counts.

Condition set 1:

1. One of the System Performance Counters is programmed to count return stack pops, event 0x36.
2. The target address of the pop is to slow level two memory.
3. The return stack pop is mispredicted, and is corrected before the target fetch has completed.

Condition set 2:

1. One of the System Performance Counters is programmed to count return stack mispredicts, event 0x37.
2. A return stack mispredict is caused by a `MOV pc, r14` or `BX r14` instruction.

Implications

The count of return stack events will be lower than the actual number of events that occurred. The performance monitor is intended to be used to help investigate code performance, and therefore is unlikely to be used by production code.

Workaround

No workaround available.

396716: Data cache clean instructions with valid RAM parity errors write to external memory

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

A data cache clean operation must read the valid RAM to determine if it should write the cache line out to external memory. If a parity error is detected on the valid RAM, then the processor cannot know if the data is valid or not. Therefore it should not write the data out to external memory, and it should cause a data abort to allow software to handle it.

This erratum causes data to still be written to external memory when there is an error on the valid RAM. The abort is taken correctly.

Conditions

1. A clean data cache line by set and way (`MCR p15, 0, Rx, c7, c10, 2`), clean and invalidate data cache line by set and way (`MCR p15, 0, Rx, c7, c14, 2`), clean entire data cache (`MCR p15, 0, Rx, c7, c10, 0`), or a clean and invalidate entire data cache (`MCR p15, 0, Rx, c7, c14, 0`) instruction is executed.
2. There is a parity error on the valid RAM of one of the accessed lines.

Implications

If a parity error occurs on the valid RAM, it may result in external memory being corrupted. However it will cause a data abort, therefore software will be aware that the error has been detected.

Workaround

This erratum may be avoided by marking all cacheable locations as write through memory. This can be done by programming the MPU memory region attributes registers, or by setting the Force Write Through bit (bit[2]) of the Cache Debug Control Register `(MCR p15,7,c15,c0,0)` .

397753: Multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1, Fixed in r0p2.

Description

A multiple word load including the PC which crosses from non Strongly-Ordered Memory to Strongly-Ordered Memory will result in two acknowledgements being passed between the LSU and the core. This leads to a loss of synchronization between the Core and LSU, and means that subsequent aborts will attributed to the wrong transactions.

This erratum occurs when a LDM including the PC crosses from a Normal or Device region of memory into a Strongly-Ordered region of memory. The same erratum occurs for a Thumb POP instruction crossing the same boundary. An optimisation in the ARM1156 causes the load of the PC to be executed first, followed by the load of the remaining registers in the original specified order. However, the acknowledgement mechanism between the core and the LSU fires twice for the operation, once from the non Strongly-Ordered Memory access, and once for the final Strongly-Ordered Memory access.

Conditions

1. Not low interrupt latency configuration.
2. An LDM or POP of at least three registers (including the PC).
3. Data alignment such that the PC and at least one other word is situated in Strongly-Ordered memory, and at least one word is situated in Normal or Device memory.

Implications

Typically Strongly-Ordered memory is allocated only for device drivers or other regions of memory which are well separated from the cached normal memory. Executing an instruction that crosses a boundary between two different types of memory region is UNPREDICTABLE. Therefore correctly written code will never stimulate the conditions for this erratum to occur.

The implication if invoked is either for abort information to be assigned to an incorrect memory operation, or for the processor to deadlock. The core expects to receive one acknowledgement for each Load or Store (including Load/Store multiple) that is issued. If it receives an additional acknowledgement, it behaves as if the next transaction has already been acknowledged, so any data abort from that transaction will be ignored, and instead be attributed to a subsequent load or store. If the synchronization between the core and the LSU gets out of synchronization by more than 4 steps, then this process can result in deadlock, as the comparison counters “wrap around”, resulting the LSU appearing to be “behind” the core.

Workaround

For correctly written code, no workaround is needed.

401148: ETM may not gain data synchronization on power up

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r0p4.

Description

To save power, the clock to the ETM is gated when it is not in use. The ETM is enabled by clearing the ETM power down bit, which is bit[0] in the ETM control register.

When the ETM is enabled, the processor can be in the middle of executing instructions. The ETM must synchronize with the processor before any trace can be output, to ensure that data values get associated with the correct instructions. Under the conditions below, the mechanism which synchronizes the ETM to the processor can behave incorrectly, which can cause the ETM to associate data addresses and values with the wrong instructions.

Conditions

Condition set 1

The following conditions must occur at the same time:

1. The power down bit in the ETM control register is cleared.
2. A previous load or store instruction is in progress.
3. A branch instruction is executed, and is folded on to the following instruction.
4. The following instruction is a load or store instruction.
5. The branch is incorrectly predicted.

Condition set 2

The following conditions must occur at the same time:

1. The power down bit in the ETM control register is cleared.
2. A previous load or store instruction is in progress.
3. A new load or store instruction is executed.
4. The new load or store instruction fails its condition codes.

Additional conditions must also be met regarding the state of the processor pipeline. This erratum might not occur even if the above conditions are met.

Implications

Data synchronization will not be achieved, so data will be associated with the wrong instruction. All data address and data value trace will be incorrect. Any comparisons based on data addresses and data values will be incorrect. Synchronization may be regained if a precise data abort occurs, or an interrupt occurs when the processor is in low interrupt latency configuration, or the processor is reset.

Workaround

The workaround available depends on the way the ETM is accessed.

1. The ETM is powered up by software running on the processor writing to the ETM memory mapped registers. The software must execute a DSB (data synchronisation barrier) instruction before and after writing to the ETM control register. Alternatively, the ETM memory region can be configured as strongly ordered memory.
2. The ETM is powered up by another component on the SoC, for example a debugger writing via the JTAG interface. The debugger must ensure that the processor is halted in debug state while the ETM control register is written to power up the ETM.

434752: FAR/FSR write immediately following precise abort can corrupt FAR/FSR

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1,r0p2,r0p3, Fixed in r0p4.

Description

A load to normal non-cacheable memory that generates an external abort updates the Data Fault Status Register and the Data Fault Address Register.

If the aborting load is followed in program order by a write to the Data Fault Status Register then the Data Fault Status Register may be corrupted.

If the aborting load is followed in program order by a write to the Data Fault Address Register then the Data Fault Address Register may be corrupted.

Conditions

1. Bit 21 of the CP15 Control Register is set
2. A load to normal non-cacheable memory generates an external abort
3. The load is immediately followed in program order by a write to the Data Fault Status Register or the Data Fault Address Register

Implications

The code sequence for stimulating this erratum is unlikely to be encountered in applications. If the write to the FSR/FAR is resetting the register to 0x0 then this erratum will not be observed.

If encountered this erratum may result in the FAR containing an address that is not associated with the external abort and the FSR containing an incorrect or invalid value.

Workaround

To prevent this erratum occurring it must be ensured that all outstanding memory accesses should have completed before writing to the FSR/FAR. This can be accomplished by inserting a data memory barrier before writing to the FAR/FSR. For example:

```
MCR p15, 0, <Rd>, c7, c10, 5 ; Data Memory Barrier
MCR p15, 0, <Rd>, c5, c0, 0  ; FSR write
```

497917: Wait For Interrupt can cause deadlock when FREEDBGTCKEN is HIGH

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Cat 3, Present in: r0p0,r0p1,r0p2,r0p3,r0p4, Open.

Description

The ARM1156T2-S and ARM1156T2F-S processor cores have a Debug Test Access Port that can be connected to a JTAG-based debugger. This interface includes a free running version of the debug clock enable, FREEDBGTCKEN. A debugger must drive this input with external synchronization logic in order to access the processor's debug features. This input is normally held low when no debugger is connected. The processor cores also have an INTSYNCEN input that can be tied high when synchronous interrupts are used. This reduces interrupt latency as the interrupt synchronizers are bypassed.

The Wait For Interrupt (WFI) operation (MCR p15,0,<Rd>,c7,c0,4) puts the processor into a low-power state and stops it executing more instructions until an interrupt (or debug) request occurs, regardless of whether the interrupts are disabled. This low-power state causes the core clock signal (CLKIN) to be gated from most of the processor core logic. When FREEDBGTCKEN is high the core clock signal is not gated because the processor core might have to perform an operation requested by the debugger. If FREEDBGTCKEN is high when an interrupt occurs after a WFI operation has been executed, the timing of the state machine that restarts the processor core is affected. This makes it possible for the processor core to deadlock when the interrupt occurs.

Conditions

The following conditions must all be met:

1. The INTSYNCEN signal must be high, which will disable the interrupt synchronization registers.
2. The FREEDBGTCKEN signal must be high. If the Debug Test Access Port is being driven by an active debugger, the FREEDBGTCKEN signal will periodically go high.
3. A WFI operation must be executed.
4. An interrupt must occur.

In addition to the above conditions, this erratum requires specific timing between internal signals and depends on the state of the instruction cache. Therefore any code that replicates the given conditions might not stimulate this erratum.

Implications

The processor core might deadlock if it is being accessed by a debugger while executing code that uses WFI. The processor core might also deadlock in a system that ties FREEDBGTCKEN high while executing code that uses WFI.

Workaround

The system workaround for this erratum is to tie INTSYNCEN low to enable the interrupt synchronization registers.

A system that does not require the JTAG interface can instead implement the second workaround which is to tie FREEDBGTCKEN low.

Errata - Implementation

381511: Pipelined reset may cause ATPG DRC errors

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	IMPL, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The ARM1156 includes a number of synchronisers and repeaters on the reset signals.

During scan insertion these flip-flops are included in the test scan chains. Two methods are provided to prevent reset pulses being generated during shift operations of the scan chains. The first involves bypassing the synchronisers by asserting the RSTBYPASS signal. This method functions correctly but there is a loss of coverage in the synchronisers. The second method relies on masking the output of the synchronisers during shift operations. Two of the synchronisers lack the masking logic which can result in a significant loss of fault coverage.

Implications

If RSTBYPASS is not asserted for scan test, spurious reset pulses will be generated by the 2 synchronisers. This will reset any D-Type flip flop connected to the synchronisers. This will prevent parts of the scan chains from being initialised during shift operations and so result in lower fault coverage.

Workaround

RSTBYPASS should be asserted for scan test.

391574: RREADY AXI signals change when ACLKEN is low

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	IMPL, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The ARM1156T2-S and ARM1156T2F-S processors use a single clock, CLKIN. To allow the AXI busses to be run at the same or lower clock frequency a clock enable signal is provided for each AXI port.

The inputs from the AXI busses are sampled when the clock enable signal ACLKENx is high. The RREADY output for the read channels (RREADYI, RREADYRW and RREADYP) may change when ACLKENx is low.

Conditions

The core to AXI clock ratio is 2:1 or greater

Implications

The timing paths from RREADYx cannot be marked as multicycle paths. Timing closure on these paths may not be possible.

Note that the erratum cannot be detected in unit delay simulations

Workaround

If timing closure on the RREADYx paths is not possible without the usage of multicycle paths, the erratum can be avoided by the insertion of AXI register slices on the read data channel of the 3 AXI interfaces (I, RW and P) of the ARM1156

Errata - Testchip

352364: ARM1156 MBIST test can fail if multiple arrays are simultaneously enabled

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Testchip, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The ARM11 Memory Built-In Self Test (MBIST) controller is implemented in the ARM1156T2(F)-S Test Chip to test the Test Chip RAMs, the ARM1156T2(F)-S RAMs and, if implemented, the ETB11 RAMs. When configuring the controller dedicated to the ARM1156T2(F)-S RAMs via its MBIST Instruction Register, it is possible to enable multiple RAMs at the same time. In this case, the arrays are tested sequentially and the number of columns is set to 4, regardless of the value set in the MBIST Instruction Register. The number of rows is normally calculated based on the given number of columns and the size of the tested RAM. Due to this erratum, the resulting number of rows can be wrongly calculated, resulting in the MBIST sequence to fail.

Conditions

1. Multiple RAMs are enabled at the same time in the ARM1156T2(F)-S MBIST Instruction Register

Implications

Under the condition described above, the MBIST sequence can fail regardless of the actual RAM test result.

Workaround

Two workarounds are possible for this erratum:

1. Only one RAM should be enabled at the any time in the MBIST Instruction Register.
2. The number of columns of the MBIST Instruction Register must be set to 4 if multiple RAMs are enabled at the same time.

In either of these two cases, the erratum does not occur.

397822: ARM1156 testchip has incorrect JTAG ID code / part number

Status

Affects:	product(s) ARM1156T2-S, ARM1156T2F-S.
Fault status:	Testchip, Present in: r0p0,r0p1, Fixed in r0p2.

Description

The JTAG TAP ID register in the ARM1156 testchip has an incorrect value. The manufacturers' part number should contain 0xF226, but actually contains 0xF21D. The incorrect value corresponds to an ARM1136 testchip.

Implications

Systems which attempt to identify the testchip using the JTAG TAP ID code will get an incorrect value. If they base their behavior on this value then it might cause unexpected results.

Workaround

The processor's device ID code can be read through the scan chain. If the testchip ID is that for an ARM1136 testchip and the processor indicates that it is an ARM1156 processor, then the testchip ID is incorrect and may be ignored.