ARM Core
ARM1136J-S™ / ARM1136JF-S™ (AT310/AT260)
Errata Notice

This document contains all errata known at the date of issue in supported releases up to and including revision r1p5 of ARM1136JF-S (AT260) and ARM1136J-S (AT310) products.

© Copyright ARM Limited 2010. All rights reserved.

Non Confidential

Proprietary notice

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

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

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

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

Document confidentiality status

This document is Non Confidential.

Web address

http://www.arm.com/

 © Copyright ARM Limited 2010. All rights reserved.

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.

© Copyright ARM Limited 2010. All rights reserved.

Contents

 © Copyright ARM Limited 2010. All rights reserved.

 © Copyright ARM Limited 2010. All rights reserved.

 © Copyright ARM Limited 2010. All rights reserved.

 © Copyright ARM Limited 2010. All rights reserved.
Non Confidential

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

 © Copyright ARM Limited 2010. All rights reserved.

Change Control

12 Jan 2010: Changes in Document v26

Page	Status	ID	Cat	Summary
132	New	724703	Impl	Peripheral Port Asynchronous clock crossing

29 Jul 2009: Changes in Document v25

Page	Status	ID	Cat	Summary
100	New	720620	Cat 2	Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line

13 Jul 2007: Changes in Document v24

Page	Status	ID	Cat	Summary
93	Updated	408660	Cat 2	VFP and GCP instructions are not executed in debug state when the J or T bit is set
121	Updated	401725	Cat 3	ETM may not gain data synchronization on power up
126	New	427336	Cat 3	FAR/FSR write immediately following precise abort can corrupt FAR/FSR
127	New	441930	Cat 3	STC immediately following precise external abort updates data cache
131	Updated	409051	Impl	Pipelined reset may cause ATPG DRC errors

13 Mar 2007: Changes in Document v23

Page	Status	ID	Cat	Summary
94	Updated	411920	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill
97	New	424067	Cat 2	An interrupted Invalidate TLB Entry on ASID Match operation can result in microTLB corruption
99	New	424865	Cat 2	Prefetch Instruction Cache Range can cause incorrect operation
121	New	401725	Cat 3	ETM may not gain data synchronization on power up
NA	Removed	415662	Cat 2	Invalidate Instruction Cache by Index might corrupt cache when used with background prefetch range

06 Feb 2007: Changes in Document v22

Page	Status	ID	Cat	Summary
26	Updated	317041	Cat 2	TTBR0/TTBR1 bits[4:3] do not read back correctly
28	Updated	317043	Cat 2	Writes to the FAR, DFSR and Data Cache Debug Control Registers can deadlock when following a Strongly-Ordered Write
39	Updated	324511	Cat 2	The contents of the IFSR may give incorrect cause of a prefetch abort.
46	Updated	326103	Cat 2	FSR write bit incorrect on a SWP to read-only memory
56	Updated	328429	Cat 2	An Associative Invalidate Instruction Cache or Prefetch operation can result in microTLB corruption
59	Updated	329839	Cat 2	In low interrupt latency mode a TLB generated abort of a V6 unaligned load or store can result in a core deadlock
73	Updated	371025	Cat 2	Associative Invalidate Instruction Cache operation can fail

 © Copyright ARM Limited 2010. All rights reserved.

88	Updated	405875	Cat 2	Interrupted Invalidate Instruction Cache by Index might corrupt cache
90	Updated	406973	Cat 2	CLREX instruction might be ignored during data cache line fill
91	Updated	408022	Cat 2	Canceled write to the Context ID register might update the ASID
94	Updated	411920	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill
NA	Updated	415662	Cat 2	Invalidate Instruction Cache by Index might corrupt cache when used with background prefetch range
118	Updated	329838	Cat 3	An exception from Java state can, under extremely rare circumstances, corrupt a CP15 operation run shortly afterwards
131	Updated	409051	Impl	Pipelined reset may cause ATPG DRC errors

12 Dec 2006: Changes in Document v21

Page	Status	ID	Cat	Summary
56	Updated	328429	Cat 2	An Associative Invalidate Instruction Cache or Prefetch operation can result in microTLB corruption
73	Updated	371025	Cat 2	Associative Invalidate Instruction Cache operation can fail
88	New	405875	Cat 2	Interrupted Invalidate Instruction Cache by Index might corrupt cache
90	Updated	406973	Cat 2	CLREX instruction might be ignored during data cache line fill
91	New	408022	Cat 2	Canceled write to the Context ID register might update the ASID
94	New	411920	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill
NA	New	415662	Cat 2	Invalidate Instruction Cache by Index might corrupt cache when used with background prefetch range
125	New	413968	Cat 3	A breakpoint on an instruction executed with the J and T bits set might take the undefined instruction trap

07 Nov 2006: Changes in Document v20

Page	Status	ID	Cat	Summary
83	Updated	399234	Cat 2	Data cache entry can be incorrectly marked as dirty
87	New	403345	Cat 2	Jazelle Security
90	New	406973	Cat 2	CLREX instruction might be ignored during data cache line fill
93	New	408660	Cat 2	VFP and GCP instructions are not executed in debug state when the T bit is set
120	Updated	397389	Cat 3	Conditional load or store instructions reading uninitialised condition codes may result in deadlock
124	New	408023	Cat 3	BTAC invalidate ignored while BTAC disabled
122	New	407121	Cat 3	Unpredictable self-modifying Jazelle code sequence can cause core to hang when BTAC is on
131	New	409051	Impl	Pipelined reset may cause ATPG DRC errors

© Copyright ARM Limited 2010. All rights reserved.

04 Aug 2006: Changes in Document v19

Page	Status	ID	Cat	Summary
76	Updated	380532	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock
78	New	395242	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction
81	New	397187	Cat 2	Interrupts from VIC interface may be taken multiple times
80	New	397185	Cat 2	VFP or GCP instruction that stalls may not trace its data with the ETM
83	New	399234	Cat 2	Write back data cache entry evicted by write through entry causes data corruption
85	New	399301	Cat 2	Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration
120	New	397389	Cat 3	Conditional load or store instructions executed after reset may deadlock

12 Apr 2006: Changes in Document v18

Page	Status	ID	Cat	Summary
76	New	380532	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock

13 Jan 2006: Changes in Document v17

Page	Status	ID	Cat	Summary
73	New	371025	Cat 2	Invalidate Instruction Cache operation can fail

26 Oct 2005: Changes in Document v16

Page	Status	ID	Cat	Summary
72	New	364296	Cat 2	Possible Cache Data Corruption with Hit-Under-Miss

02 Aug 2005: Changes in Document v15

Page	Status	ID	Cat	Summary
22	Updated	351912	Cat 1	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)
24	Updated	353494	Cat 1	Rare conditions can cause a corruption of the Instruction Cache
70	Updated	351914	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV PC)

08 Jul 2005: Changes in Document v14

Page	Status	ID	Cat	Summary
22	Updated	351912	Cat 1	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)
24	New	353494	Cat 1	Rare conditions can cause a corruption of the Instruction Cache
70	Updated	351914	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV PC)

 © Copyright ARM Limited 2010. All rights reserved.

15 Jun 2005: Changes in Document v13

Page	Status	ID	Cat	Summary
22	Updated	351912	Cat 1	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)
70	Updated	351914	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV to PC)

13 Jun 2005: Changes in Document v12

Page	Status	ID	Cat	Summary
22	New	351912	Cat 1	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)
69	New	349888	Cat 2	Enabling the Cache Size Restriction can result in a deadlock
70	New	351914	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV to PC)
130	New	351947	Impl	Instruction Cache size of 4 or 8 kB can cause deadlock

07 Apr 2005: Changes in Document v11

Page	Status	ID	Cat	Summary
27	Updated	317042	Cat 2	An External Abort on a single word during a Multiple Word Load to the Peripheral Port can result in Deadlock
32	Updated	322139	Cat 2	External Aborts on TLB Page Table Walks are not properly reported
33	Updated	324501	Cat 2	Interrupts received when changing interrupt masks in low interrupt latency configuration can cause incorrect operation.
52	Updated	328409	Cat 2	Externally aborted loads to the PC can result in deadlock
67	New	343942	Cat 2	VFP load/store multiple instructions of length > 16 words that cross memory regions can result in a deadlock
68	New	344158	Cat 2	Externally aborted LDM to the PC might not take data abort exception
102	Updated	303162	Cat 3	Supersections can return incorrect physical address when subpages enabled
106	Updated	303164	Cat 3	Aborts on the first part of a V6 unaligned access to strongly ordered memory may not be reported.
116	Updated	328431	Cat 3	VFP can take an Inexact exception on non-CDP VFP instructions
119	New	344292	Cat 3	Corrupting the TLB can deadlock the processor
128	New	344291	Impl	RAM bit and byte write enables are incorrectly driven

27 Jan 2005: Changes in Document v10

Page	Status	ID	Cat	Summary
49	Updated	326296	Cat 2	Potential incorrect operation of Clean Entire / Clean and Invalidate Entire Data Cache operations in low latency mode
60	New	333030	Cat 2	The GE flags might not be updated

 © Copyright ARM Limited 2010. All rights reserved.

61	New	334358	Cat 2	Mispredicted branch folded onto an MCR can be missed from ETM trace
62	New	334360	Cat 2	A coprocessor instruction that stalls and then is cancelled by an interrupt loses ETM trace sync
63	New	335567	Cat 2	STREX instruction might return incorrect status
65	New	336501	Cat 2	Wait For Interrupt does not drain the Data Cache write buffer
66	New	336509	Cat 2	Disabling the loading of the main TLB does not prevent updates to the lockdown region

11 Aug 2004: Changes in Document v9

Page	Status	ID	Cat	Summary
59	New	329839	Cat 2	In low interrupt latency mode a TLB generated abort of a V6 unaligned load or store can result in a core deadlock
118	New	329838	Cat 3	An exception from Java state can, under extremely rare circumstances, corrupt a CP15 operation run shortly afterwards
117	New	329837	Cat 3	CP15 Instruction Cache Data RAM Read Operation reads only even addresses

12 Jul 2004: Changes in Document v8

Page	Status	ID	Cat	Summary
51	New	328395	Cat 2	In Low Interrupt Latency configuration, an interrupt to a SWP on to non-cachable memory can result in a system deadlock
52	New	328409	Cat 2	Externally aborted loads to the PC can result in deadlock
54	New	328428	Cat 2	External abort on the non requested word of a cache linefill can lead to cache corruption
56	New	328429	Cat 2	An instruction cache invalidate by MVA or Prefetch by MVA operation, if interrupted, can result in microTLB corruption
58	New	328430	Cat 2	A precise external data abort followed immediately by a TLB generated data abort can result in an incorrect DFSR and FAR
116	New	328431	Cat 3	[AT260_ARM1136JF-S] VFP can take an Inexact exception on non-CDP VFP instructions

08 Jun 2004: Changes in Document v7

Page	Status	ID	Cat	Summary
46	New	326103	Cat 2	FSR write bit incorrect on a SWP to read-only memory
48	New	326295	Cat 2	EDBGRQ does not cause Debug halting when Debug is not enabled
49	New	326296	Cat 2	Low interrupt latency mode problem with Cache Clean All / Clean and Invalidate All operations

21 May 2004: Changes in Document v6

Page	Status	ID	Cat	Summary
40	New	324512	Cat 2	A V6 unaligned store to the first two words of a cache line in a write-through memory region can deadlock the ARM1136

 © Copyright ARM Limited 2010. All rights reserved.

42	New	324513	Cat 2	PLDs may incorrectly report external aborts
43	New	324514	Cat 2	Interworking between instruction TCM and instruction Cache can result in incorrect instruction execution
45	New	325157	Cat 2	Associative ICache maintenance operations can deadlock
114	Updated	324519	Cat 3	Possibility of ETM not correctly synchronising instruction and data with processor core at ETM power up.
115	New	325158	Cat 3	FSR content may be incorrectly updated if an interrupt occurs immediately before an aborting load or store

4 May 2004: Changes in Document v5

Page	Status	ID	Cat	Summary
32	New	322139	Cat 2	External Aborts on TLB Page Table Walks are not properly reported
33	New	324501	Cat 2	Interrupts received when changing interrupt masks in low interrupt latency configuration can cause incorrect operation.
35	New	324508	Cat 2	FAR Can Load An Incorrect Address by an External Imprecise Abort on a Load Occurring at the same time as Page Fault
37	New	324509	Cat 2	A multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory
38	New	324510	Cat 2	Conditional BXJ instruction will not execute if immediately after a flag setting operation
39	New	324511	Cat 2	The contents of the IFSR may give incorrect cause of a prefetch abort.
112	New	324515	Cat 3	Vector catch on vector 0x18 while VIC enabled
113	New	324518	Cat 3	Coprocessors only have user mode privileges until the first branch or mode change instruction.
114	New	324519	Cat 3	Possibility of ETM not correctly synchronising instruction and data with processor core at ETM power up.

Changes in Document v4 – N/A

No changes.

9 Dec 2003: Changes in Document v3

Page	Status	ID	Cat	Summary
27	New	317042	Cat 2	An External Abort on a single word during a Multiple Word Load to the Peripheral Port can result in Deadlock
28	New	317043	Cat 2	Writes to the FAR, DFSR and Data Cache Debug Control Registers can deadlock when following a Strongly-Ordered Write
30	New	317044	Cat 2	A Hardware Breakpoint on a VFP instruction can result in incorrect operation when closely followed by a GCP instruction
26	New	317041	Cat 2	TTBR0/TTBR1 bits[4:3] do not read back correctly
110	New	317047	Cat 3	ACPFLUSH may flush instructions in a Generic Coprocessor after that coprocessor has been deselected

 © Copyright ARM Limited 2010. All rights reserved.

111	New	317048	Cat 3	Performance Counters May Fail to Interrupt on Roll-over of Double Incrementing Events
109	New	317046	Cat 3	Debug DSCR Sticky Bits are cleared when the DSCR is written from Scan
108	New	317045	Cat 3	Bits [9:8] of the Data FSR can be written to 1

18 Aug 2003: Changes in Document v2

Page	Status	ID	Cat	Summary
106	New	303164	Cat 3	Aborts on the first part of a V6 unaligned access to strongly ordered memory may not be reported.

23 Jul 2003: Changes in Document v1

Page	Status	ID	Cat	Summary
102	New	303162	Cat 3	Supersections can return incorrect physical address when subpages enabled
104	New	303163	Cat 3	ETM: A low latency interrupt on a java conditional branch may be incorrectly traced.

 © Copyright ARM Limited 2010. All rights reserved.

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.

A NOTE ON VERSION r0p4:

An r0p4 is a patched r0p2 version of the ARM1136 with both category 1 errata fixed:

353494: Rare conditions can cause a corruption of the Instruction Cache

351912 The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)

All other category 2 and category 3 errata are the same as the r0p2 version.

A NOTE ON VERSION r0p5:

An r0p5 is a patched r0p2 version of the ARM1136 which has the same two errata fixed as in an r0p4 version plus the following additional erratum fixed:

364296: Possible Cache Data Corruption with Hit-Under-Miss

All other category 2 and category 3 errata are the same as the r0p2 version.

A NOTE ON VERSION r0p6:

An r0p6 is a patched r0p2 version of the ARM1136 which has the same three errata fixed as in an r0p5 version plus the following additional erratum fixed:

380532: Load Multiple to the PC can be corrupted or result in deadlock

All other category 2 and category 3 errata are the same as the r0p2 version.

A NOTE ON VERSION r0p7:

An r0p7 is a patched r0p2 version of the ARM1136 which has the same four errata fixed as in an r0p6 version plus the following additional erratum fixed:

399234 "Data cache entry can be incorrectly marked as dirty"

All other category 2 and category 3 errata are the same as the r0p2 version.

ID	Cat	Summary of Errata for Metal Fix Rev 0 Patch Releases (All other category 2 and category 3 errata are the same as r0p2)	r0p4	r0p5	r0p6	r0p7
351912	Cat 1	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)				
353494	Cat 1	Rare conditions can cause a corruption of the Instruction Cache				
364296	Cat 2	Possible Cache Data Corruption with Hit-Under-Miss	X			
380532	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock	X	X		
399234	Cat 2	Data cache entry can be incorrectly marked as dirty	X	X	X	

 © Copyright ARM Limited 2010. All rights reserved.

ID	Cat	Summary of Erratum	r0p2	r1p0	r1p1	r1p2	r1p3	r1p4	r1p5
344291	Impl	RAM bit and byte write enables are incorrectly driven	X						
351947	Impl	Instruction Cache size of 4 or 8 kB can cause deadlock		X	X	X	X	X	
409051	Impl	Pipelined reset may cause ATPG DRC errors	X	X	X	X	X	X	
724703	Impl	Peripheral Port Asynchronous clock crossing	X	X	X	X	X	X	X
351912	Cat 1	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)	X						
353494	Cat 1	Rare conditions can cause a corruption of the Instruction Cache	X	X					
317041	Cat 2	TTBR0/TTBR1 bits[4:3] do not read back correctly	X						
317042	Cat 2	An External Abort on a single word during a Multiple Word Load to the Peripheral Port can result in Deadlock	X						
317043	Cat 2	Writes to the FAR, DFSR and Data Cache Debug Control Registers can deadlock when following a Strongly-Ordered Write	X						
317044	Cat 2	A Hardware Breakpoint on a VFP instruction can result in incorrect operation when closely followed by a GCP instruction	X						
322139	Cat 2	External Aborts on TLB Page Table Walks are not properly reported	X						
324501	Cat 2	Interrupts received when changing interrupt masks in low interrupt latency configuration can cause incorrect operation.	X						
324508	Cat 2	FAR can load an incorrect address if an external imprecise abort on a load occurs at the same time as a page fault	X						
324509	Cat 2	Multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory	X						
324510	Cat 2	Conditional BXJ instruction will not execute if immediately after a flag setting operation	X						
324511	Cat 2	The contents of the IFSR may give incorrect cause of a prefetch abort.	X						
324512	Cat 2	A V6 unaligned store to the first two words of a cache line in a write-through memory region can deadlock the ARM1136	X						

© Copyright ARM Limited 2010. All rights reserved.

ID	Cat	Summary of Erratum	r0p2	r1p0	r1p1	r1p2	r1p3	r1p4	r1p5
324513	Cat 2	PLDs may incorrectly report external aborts	X						
324514	Cat 2	Interworking between instruction TCM and instruction Cache can result in incorrect instruction execution	X						
325157	Cat 2	Associative ICache maintenance operations can deadlock	X						
326103	Cat 2	FSR write bit incorrect on a SWP to read-only memory	X						
326295	Cat 2	EDBGRQ does not cause Debug halting when Debug is not enabled	X						
326296	Cat 2	Potential incorrect operation of Clean Entire / Clean and Invalidate Entire Data Cache operations in low latency mode	X						
328395	Cat 2	In Low Interrupt Latency configuration, an interrupt to a SWP on to non-cachable memory can result in a system deadlock	X						
328409	Cat 2	Externally aborted loads to the PC can result in deadlock	X						
328428	Cat 2	External abort on the non requested word of a cache linefill can lead to cache corruption	X						
328429	Cat 2	An Associative Invalidate Instruction Cache or Prefetch operation can result in microTLB corruption	X						
328430	Cat 2	A precise external data abort followed immediately by a TLB generated data abort can result in an incorrect DFSR and FAR	X						
329839	Cat 2	In low interrupt latency mode a TLB generated abort of a V6 unaligned load or store can result in a core deadlock	X						
333030	Cat 2	The GE flags might not be updated	X						
334358	Cat 2	Mispredicted branch folded onto an MCR can be missed from ETM trace	X						
334360	Cat 2	A coprocessor instruction that stalls and then is cancelled by an interrupt loses ETM trace sync	X						
335567	Cat 2	STREX{B|H|D} instructions might return incorrect status	X						
336501	Cat 2	Wait For Interrupt does not drain the Data Cache write buffer	X						
336509	Cat 2	Disabling the loading of the main TLB does not prevent updates to the lockdown region	X						

© Copyright ARM Limited 2010. All rights reserved.

ID	Cat	Summary of Erratum	r0p2	r1p0	r1p1	r1p2	r1p3	r1p4	r1p5
343942	Cat 2	VFP load/store multiple instructions of length > 16 words that cross memory regions can result in a deadlock	X						
344158	Cat 2	Externally aborted LDM to the PC might not take data abort exception	X						
349888	Cat 2	Enabling the Cache Size Restriction can result in a deadlock		X	X	X	X	X	
351914	Cat 2	The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV PC)	X	X	X	X	X	X	
364296	Cat 2	Possible Cache Data Corruption with Hit-Under-Miss	X						
371025	Cat 2	Associative Invalidate Instruction Cache operation can fail	X	X	X	X	X	X	
380532	Cat 2	Load Multiple to the PC can be corrupted or result in deadlock	X	X	X				
395242	Cat 2	VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction	X	X	X	X	X	X	
397185	Cat 2	VFP or GCP instruction that stalls may not trace its data with the ETM	X	X	X	X	X	X	
397187	Cat 2	Interrupts from VIC interface may be taken multiple times	X	X	X	X	X	X	
399234	Cat 2	Data cache entry can be incorrectly marked as dirty	X	X	X	X			
399301	Cat 2	Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration	X	X	X	X	X	X	
403345	Cat 2	Jazelle Security	X	X	X	X	X	X	
405875	Cat 2	Interrupted Invalidate Instruction Cache by Index might corrupt cache	X	X	X	X	X		
406973	Cat 2	CLREX instruction might be ignored during data cache line fill		X	X	X	X	X	
408022	Cat 2	Canceled write to the Context ID register might update the ASID	X	X	X	X	X		
408660	Cat 2	VFP and GCP instructions are not executed in debug state when the J or T bit is set	X	X	X	X	X	X	
411920	Cat 2	Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill	X	X	X	X	X		

© Copyright ARM Limited 2010. All rights reserved.

ID	Cat	Summary of Erratum	r0p2	r1p0	r1p1	r1p2	r1p3	r1p4	r1p5
424067	Cat 2	An interrupted Invalidate TLB Entry on ASID Match operation can result in microTLB corruption	X	X	X	X	X	X	
424865	Cat 2	Prefetch Instruction Cache Range can cause incorrect operation	X	X	X	X	X	X	
720620	Cat 2	Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line	X	X	X	X	X	X	X
303162	Cat 3	Supersections can return incorrect physical address when subpages enabled	X						
303163	Cat 3	ETM: A low latency interrupt on a java conditional branch may be incorrectly traced.	X						
303164	Cat 3	Aborts on the first part of a V6 unaligned access to strongly ordered memory may not be reported.	X						
317045	Cat 3	Bits [9:8] of the Data FSR can be written to 1	X						
317046	Cat 3	Debug DSCR Sticky Bits are cleared when the DSCR is written from Scan	X						
317047	Cat 3	ACPFLUSH may flush instructions in a Generic Coprocessor after that coprocessor has been deselected	X						
317048	Cat 3	Performance Counters May Fail to Interrupt on Roll-over of Double Incrementing Events	X						
324515	Cat 3	Vector catch on vector 0x18 while VIC enabled	X						
324518	Cat 3	Coprocessors only have user mode privileges until the first branch or mode change instruction.	X						
324519	Cat 3	Possibility of ETM not correctly synchronising instruction and data with processor core at ETM power up.	X						
325158	Cat 3	FSR content may be incorrectly updated if an interrupt occurs immediately before an aborting load or store	X						
328431	Cat 3	VFP can take an Inexact exception on non-CDP VFP instructions	X	X	X	X	X	X	X
329837	Cat 3	CP15 Instruction Cache Data RAM Read Operation reads only even addresses	X						
329838	Cat 3	An exception from Java state can, under extremely rare circumstances, corrupt a CP15 operation run shortly afterwards	X						
344292	Cat 3	Corrupting the TLB can deadlock the processor	X						

© Copyright ARM Limited 2010. All rights reserved.

ID	Cat	Summary of Erratum	r0p2	r1p0	r1p1	r1p2	r1p3	r1p4	r1p5
397389	Cat 3	Conditional load or store instructions reading uninitialised condition codes may result in deadlock	X	X	X	X	X	X	
401725	Cat 3	ETM may not gain data synchronization on power up	X	X	X	X	X	X	
407121	Cat 3	Unpredictable self-modifying Jazelle code sequence can cause core to hang when BTAC is on	X	X	X	X	X	X	X
408023	Cat 3	BTAC invalidate ignored while BTAC disabled	X	X	X	X	X	X	
413968	Cat 3	A breakpoint on an instruction executed with the J and T bits set might take the undefined instruction trap	X	X	X	X	X	X	X
427336	Cat 3	FAR/FSR write immediately following precise abort can corrupt FAR/FSR	X	X	X	X	X	X	
441930	Cat 3	STC immediately following precise external abort updates data cache	X	X	X	X	X	X	

© Copyright ARM Limited 2010. All rights reserved.

Errata - Category 1

351912: The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)

Status

Affects: product ARM1136JF-S.

Fault status: Cat 1, Present in: r0p2, Fixed in r1p0.

Description

Also fixed in r0p4.

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

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 is 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

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 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

 © Copyright ARM Limited 2010. All rights reserved.

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.

Workaround

For the ARM1136JF-S r0p2, no software workaround exists to prevent branch folding. However, branch folding is expected to happen extremely rarely in combination with the rest of the factors listed in the conditions section. In systems where very rare double precision floating point errors cannot be tolerated, one of the following three workarounds must be employed.

1. Since branch folding (and therefore branch prediction) is required for this erratum to occur, a possible workaround is to disable branch prediction. However, this will have a severe impact on the performance of the processor and should only be contemplated where very rare floating point errors cannot be tolerated and one of the other workarounds cannot be used.
2. Apply a metal fix to the r0p2 ARM136JF-S implementation to permanently disable branch folding. Please contact ARM (support-cores@arm.com) for further information.
3. Upgrade the product to use the r1p0 version of the ARM1136JF-S which does not exhibit this problem.
4. For a 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.
5. For hand written VFP assembler code, it is possible to add a non-floating point instruction after the branch and at the destination of the branch. This will ensure that the branch is not folded onto a floating point instruction.

See also the related erratum:

351914: The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV to PC)

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

353494: Rare conditions can cause a corruption of the Instruction Cache

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 1, Present in: r0p2,r1p0, Fixed in r1p1.

Description

Also fixed in r0p4.

The ARM1136J(F)-S design includes logic to avoid the same cache line being inserted into the cache in multiple places. This logic detects if there is a possibility that an instruction cache read could come from a linefill currently being processed. It does this by testing if the cache index for the read is the same as for the linefill.

If the cache indexes are the same then the instruction cache read is re-attempted once the linefill has progressed far enough to write the required instruction into the cache. This will either generate a new cache linefill or return the required instruction.

If the instruction cache read is to a location which hits in the cache but generates an instruction TLB abort then the prefetch unit should stop fetching immediately until the abort is processed.

Due to an error in the logic for detecting cache reads to currently refilling lines, in combination with privilege faults on the cache read, it is possible to cause an instruction cache corruption.

The effect will be for data from an instruction cache linefill to be written into multiple cache ways, rather than just to the destination way for that data.

Conditions

1. An instruction TLB abort on an area of instruction memory fetched as a result of a change in program flow. This means that the memory access is triggered as the result of any instruction that changes program flow. For example a branch or MOV PC,r14. Note that this abort could ccur
2. due to a speculative instruction fetch, and hence the instruction abort may not be seen in an instruction trace.
3. One of the 4 instructions that is fetched (at the aborting region) must contain a branch that has been predicted as taken by the dynamic branch predictor. This means that this piece of code must have been run (without generating an abort) previously. This is possible as a result of a permission failure due to accessing privileged code when in user mode.
4. There must be a linefill occurring in the background which is to the same cache index as the instructions at the aborting region, but to a different physical address. The cache index is determined by bits [n:5]. Where n is a function of the cache size. For a 16K cache n is 11.

It should be noted that particular timings of both the cache lookups and the arrival of the data from the linefill request are 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.

Implications

It should be emphasised that the conditions to generate this errata are extremely rare. However, when triggered this erratum will cause the instruction cache to become corrupted. The effect is that instruction linefill data overwrites the entry at the same index in multiple cache ways. This will lead to incorrect program execution.

 © Copyright ARM Limited 2010. All rights reserved.

This erratum does not affect data cache operation.

Workaround

1. Disable branch prediction. This will impact the overall performance of the device.
2. Modify the operating system to flush the BTAC on all switches from any privileged mode into user mode and on disabling access to pages that had previously been accessible.
3. ARM is working on a potential ECO fix for this erratum. It will involve the addition of a NAND gate and one or two inverters. Please contact ARM for details of this fix.

 © Copyright ARM Limited 2010. All rights reserved.

Errata - Category 2

317041: TTBR0/TTBR1 bits[4:3] do not read back correctly

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

Bits [4:3] of Translation Table Base address registers (TTBR0, TTBR1) do not read back correctly, but instead always return 0.

The Translation Table Base address registers (TTBR0, TTBR1) use bits [4:3] to encode the outer cachable attributes that are presented on the AHB bus when a page table walk occurs. These values are encoded as

- b00 Outer NonCachable
- b01 Outer Cachable WriteBack, Write Allocate
- b10 Outer Cachable WriteThrough, No Write Allocate
- b11 Outer Cachable WriteBack, No WriteAllocate

The value that is written to these bits of these registers is correctly used to determine the HPROT[4:2] attributes on a page table walk. However, due to this errata, reading back the contents of these registers returns b00 in bit positions[4:3], regardless of the encoding that has been written to these bit positions.

Conditions

1. Bits[4:3] of the TTBR0 or TTBR1 are programmed to values other than b00
2. A read back of these registers using MRC p15, 0, <Rd>, c2, 0, 0 or MRC `p15, 0, <Rd>, c2, 0, 1`

Implications

Operating systems which are reading back the current conditions for the TTBR0/TTBR1 registers will see these regions as being non-cachable. This is most likely to occur where the TTBR registers (particularly TTBR1) are being saved on a context switch – a save and restore of these registers will result in a corruption of these bits.

Any operating system which is changing the outer cachability information for the page tables and is relying on storing the current cachability information in this register, and reading it back correctly will see incorrect values.

Practically this erratum is expected to only have consequence in systems with a Level 2 cache.

At present, Linux (rev 2.4.x, and rev 2.5.40) does not rely on the ability to read these values.

Workaround

Any code which is responsible for saving and restoring the TTBR registers needs to substitute in the correct value for these bits, rather than relying on correct operation on a save and restore of the register. This implies that the current value should be held in a separate storage structure.

 © Copyright ARM Limited 2010. All rights reserved.

317042: An External Abort on a single word during a Multiple Word Load to the Peripheral Port can result in Deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

An External Abort to a word during an multiple word load (ie LDM or LDRD, or their thumb equivalents) to the Peripheral Port, where the address of that word has bit[2] =0, and is not the final word of the multiple transaction can lead to the memory system deadlocking, unless the subsequent access (where bit[2] of the address is 1) also returns an abort.

This erratum arises due to the splitting of aligned 64 bit accesses, which is the internal representation within the ARM1136, to 2 32 bit transactions. The assumption has been made that the external abort will be returned for both words of the aligned 64 bit access. Where an external abort is returned only for the first 32 bit transaction, and the second 32 bit transaction does not return an abort, then the peripheral port enters an illegal state, and deadlocks.

Conditions

1. A multiple word load to the Peripheral Port.
2. An abort (HERROR) returned on a word with address bit [2] being 0, where this is not the final word of the burst.
3. An abort not returned on the subsequent memory transaction.

Implications

In practical cases, structural errors, such as an access to undefined region of memory typically return aborts for all words in that memory region, and such regions are aligned to boundaries greater than one word in size, so avoiding this problem. External aborts are typically returned for individual words of a multiple transaction where a parity error or other data related error is detected, and these are unlikely to be the case for accesses using the peripheral port.

However, there could be cases where aborts may be returned on individual words for other reasons, so giving a potential for a system deadlock.

Workaround

Where the peripheral port is being used, only single memory transactions should be used if there is a danger of an abort being returned on an individual word.

The Java stack should not be placed in space mapped to the peripheral port if there is a danger of an abort being returned on an individual word.

 © Copyright ARM Limited 2010. All rights reserved.

317043: Writes to the FAR, DFSR and Data Cache Debug Control Registers can deadlock when following a Strongly-Ordered Write

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

Writing to the Fault Address Register, Data Fault Status Register and Cache Debug registers when following a write to strongly-ordered memory, can result in the memory system deadlocking.

The FAR, DFSR and Data Cache Debug Control Register are written by an LSU slot when it first enters the external state, and the slot should then retire. Due to this erratum, the slot can read a flag for it not to retire if the transaction that used that entry in the external access queue previously was a Strongly Ordered access. This results in the slot being held in the external state until a subsequent access causes the state of the external access queue to be updated, by a subsequent access accessing the cache.

If the operation which follows the write which is held waits for the LSU to be empty, or waits for an available slot, then this subsequent memory access doesn't occur, and the system deadlocks. Typically CP15 operations wait for the LSU to be empty before making progress, and are a good example of conditions which can deadlock.

A DrainWriteBuffer CP15 operation is guaranteed to clear the flags for retirement, and to cause a clearing of any slots which are held wrongly in the external state.

When running in Low-Interrupt configuration, the retirement conditions of the LSU slots are different, and do not suffer from this errata.

Conditions

1. The system is running with Low Interrupt Configuration disabled.
2. A write to strongly-ordered memory is made.
3. A Write to the FAR, Data FSR or the Data Cache Debug registers is performed within three memory accesses of the write to strongly-ordered memory.
4. The subsequent memory access relies on the slot being retired before making progress.

Implications

Writes to the FAR, Data FSR or Data Cache Debug register can appear to sporadically cause system deadlocks, typically when followed by a CP15 Operation.

Writes to the FAR, DataFSR and Data Cache Debug register are not used by user code as they are privileged only operations, and are expected to be incorporated as part of a Hardware Abstraction Layer of an Operating System.

For Linux (rev 2.4.x or 2.5.40), these operations are not performed, and generally it is not a function that is typically required within an Operating System.

Debuggers are expected to write to these registers, and need to implement the DrainWriteBuffer workaround.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

Following every Write to the FAR, Data FSR or Data Cache debug register should be followed by a DrainWriteBuffer operation before any other memory operation is performed.

 © Copyright ARM Limited 2010. All rights reserved.

317044: A Hardware Breakpoint on a VFP instruction can result in incorrect operation when closely followed by a GCP instruction

Status

Affects: product ARM1136JF-S.

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

Description

A hardware breakpoint on a VFP instruction could trigger incorrect operation including potentially a pipeline deadlock when closely followed by a GCP coprocessor instruction.

A hardware breakpoint on a coprocessor instruction should, depending on whether the processor is configured for halt mode or monitor mode debug, trigger halt mode debug entry, or entry into abort mode. The coprocessor instruction itself should not be executed.

If a coprocessor instruction to a different coprocessor number that the currently selected coprocessor is executed, the processor will disable the currently selected coprocessor and enable the new coprocessor.

Due to this errata a VFP instruction which has a hardware breakpoint set on it, when followed closely by a GCP coprocessor instruction, will not be correctly flushed from the VFP. This occurs because the VFP is disabled before the VFP instruction has been flushed due to the breakpoint, from the VFP. When a following VFP instruction is executed the processor may execute incorrectly, and may potentially deadlock.

Conditions:

1. A VFP instruction with a hardware breakpoint set on it.
2. The VFP instruction is either: a) Directly followed by a GCP coprocessor instruction, or b) Followed by a GCP coprocessor instruction with a single intermediate ARM instruction in between. In this second case the breakpointed VFP instruction must also stall in the ARM pipeline during execution.

Note that the second situation may be stimulated by a code sequence similar to:

```
   Fxxx
   B skip_literal_pool
   <literal pool data> ; literal pool data represents a GCP instruction.
skip_literal_pool
      .....
```

Implications

Unless debugging this erratum will have no effect.

When debugging code it is unlikely this erratum will be triggered, since it requires a GCP instruction to occur either directly after a VFP instruction, or with one ARM instruction in between.

The use of software breakpoints cannot trigger this erratum.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

If GCP instructions are not used, or VFP instructions are not placed directly before a literal pool whose first word of data appears to be a GCP instruction (ie has bits[27:26]=11 and bits[11:8] != 0xA or 0xB) no workaround is necessary.

If these conditions cannot be used, a software breakpoint should be used in place of a hardware breakpoint. RVD offers the ability to select whether a software breakpoint or a hardware breakpoint is used when setting the breakpoint. For AXD, the environment variable

```
$sw_breakpoints_preferred
```

can be set to 1 to use software breakpoints.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

322139: External Aborts on TLB Page Table Walks are not properly reported

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

If an External Abort is returned on a page table walk at the same time as the final cycle of a Drain Write Buffer or Data Memory Barrier instruction, the external abort is not reported as a fault in the page tables.

The mechanism for performing a Drain Write Buffer or Data Memory Barrier instruction involves a mechanism which is treated within the data-side memory system of the ARM1136 in a very similar manner to a Strongly-Ordered read. However, no such read transaction is generated, and a signal is asserted at this time to force the acknowledgement on the LSU interface and to ensure no spurious abort is generated. However, this signal causes the Page Table Walk that is returning on this cycle to have its error masked.

The data returned on the page table walk is returned to the core, so if the data returned is forced to 0 when there is an external abort, the effect will be to see a translation fault.

Conditions

1. Page Table Walks are generating an External Abort
2. The External Abort is retuned in precisely the same cycle that a Drain Write Buffer takes effect.

Implications

External Aborts on page table walks are not expected to occur in most systems, and when they do occur, they are likely to be system fatal as the operating systems do not make use of external aborts on page tables for functional reasons.

If the page tables are set up to point to a region of aborting memory then the External Abort can be missed, and whatever data is returned with the external abort will be used as a legitimate page table entry. If the external abort is being asserted as part of a security mechanism, and the data being returned in the aborting case, then a security issue can arise.

It should be noted that the page table walk associated with this errata can only arise from an instruction fetch or a DMA transaction.

Workaround

Ensuring that aborting regions return a value of 0 will cause this abort to be transformed into a Translation Fault, so avoiding any security implications from picking up a valid page table descriptor.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

324501: Interrupts received when changing interrupt masks in low interrupt latency configuration can cause incorrect operation.

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

When in low interrupt latency configuration, an interrupt received just as interrupts are being masked out can cause any Load Multiple or multiple loads following the CPSID instruction to deadlock or return wrong data.

This errata requires the ARM1136 to be operating in low interrupt latency configuration.

When the ARM1136 receives a CPSID instruction to mask out interrupts, any interrupt received at the time that the CPSID is executed will result in a pause in the execution of the instructions to allow for pipelining delays in changing the interrupt masks to be resolved. This pause can temporarily stall a multi-cycle load operations (ARM LDM or LDRD; Thumb POP) between the first and second transactions of the load. When the stall is resumed, the access which was paused can be presented to the cache incorrectly.

Conditions

1. Low Interrupt Latency Configuration

2. Any of the following code sequences. Note that the CPSID instruction is masking out interrupts. (If a mode change is involved then this errata will not occur). All the instructions must be cached so there is no delay between executing the CPSID and the following instruction.

```
CPSID  i
LDMIA  r13,[r0-rx]
```

Or:

```
CPSID  i
LDRD   r13, r0              ; unaligned
LDR    r13, r1
```

Or in THUMB mode:

```
CPSID  i
POP    [r0-rx]       ;
```

3. An interrupt arriving at the last possible moment before it is masked out internally.

Implications

Any code which masks out interrupts can possibly corrupt or deadlock the ARM1136 if it is followed immediately by a multi-cycle load operation to a cacheable region and an interrupt occurs at a particular moment.

The instruction CPSID is not generated by the compiler so modifications will only have to be made to code where this has been explicitly inserted.

Workaround

Any time the CPSID instruction is used it must be followed by a NOP to allow the register to settle before executing any following instructions.

 © Copyright ARM Limited 2010. All rights reserved.

An alternative workaround can be invoked which increases the worst case interrupt latency to that which exists when the low interrupt configuration is disabled, but keeps the precise nature of external read aborts. This workaround involves setting the undocumented configuration bit in Auxiliary Control Register bit [31] to 1. This bit was designed to disable the mechanism which is in error, but to keep the external abort behaviour of having precise aborts.

 © Copyright ARM Limited 2010. All rights reserved.

324508: FAR can load an incorrect address if an external imprecise abort on a load occurs at the same time as a page fault

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

The Fault Address Register (FAR) can be loaded with an incorrect address by an external imprecise abort on a Load being returned by the memory sub-system at the same time as a page fault occurs.

This erratum occurs when a load miss returns an imprecise data abort to the Load-Store Unit (LSU) the clock cycle after a precise data abort has been reported by the TLB. The Imprecise abort will have been generated by a load which appeared in program order before the load or store which generates the precise data abort.

The ARM1136 will respond initially to the precise external abort, but the FAR of this abort will be the OR of the virtual address of the load or store that caused the TLB generated abort and the VA of the imprecisely aborting load.

Conditions

1. Not low interrupt latency configuration
2. A load is issued which misses the Data cache and makes an external access which receives a data abort.
3. Load or Store has a data abort generated by the TLB

Implications

This erratum will only occur when there has been what is, in most systems, an unrecoverable system error, and the net effect is that an unrecoverable system error will be reported, but may be reported with incorrect attributes.

The Load or Store which receives an abort from the TLB will have its VA corrupted as it is loaded into the FAR thereby resulting in an incorrect address being presented to the abort handler for the precise data abort. Such TLB aborts will either be correctable errors (such as page faults) or unrecoverable errors:

1. If the TLB abort is an unrecoverable error, then the address that caused the unrecoverable error will be incorrectly reported. Since imprecise external aborts on loads are expected to be unrecoverable errors, this leads to an occasional inaccuracy in reporting of unrecoverable errors.
2. If the TLB abort is a correctable error, the operating system will attempt to correct the error for the erroneous address. This will result in one of two conceivable outcomes:

- A legal page other than the one that faulted will be mapped, which leads to a minor inefficiency in the use of memory. On returning from the abort, the imprecise abort will be recognized, so causing the correct unrecoverable error.
- The operating system will detect a fault on a page which should not fault. This is likely to lead to an unrecoverable OS error, which means that the result is an occasional inaccuracy in reporting the source of unrecoverable errors.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

Executing in a system where there can be no imprecise external aborts will result in this erratum not being seen.

 © Copyright ARM Limited 2010. All rights reserved.

324509: Multiple word load including the PC that crosses from non Strongly-Ordered to Strongly-Ordered Memory

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

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 ARM1136 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

In all operating systems which have been investigated, it is not expected that adjacent virtual addresses will be assigned to Normal (or Device) Memory and Strongly-Ordered Memory, so the conditions for this erratum to occur are not met. Typically Strongly-Ordered memory is allocated only for device drivers or other regions of memory which are well separated from the cached normal memory.

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 acknowledgement, 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

It is not expected that code will include LDM or POP instructions which cross from normal into strongly ordered memory. However configuring the design to function in the low interrupt latency configuration will ensure this erratum is avoided.

 © Copyright ARM Limited 2010. All rights reserved.

324510: Conditional BXJ instruction will not execute if immediately after a flag setting operation

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

A BXJ instruction is used by Java Virtual Machines to enter Java state. If a conditional BXJ instruction is executed directly after an instruction which sets the condition codes it will fail to take the branch into java state.

Conditions

1. An instruction which sets the condition codes. e.g. CMP r1,r2
2. Directly following this instruction is a conditional BXJ instruction. e.g. BXJNE r5

Implications

Under the conditions described above the BXJ instruction fails to execute, as if it had failed its condition code check.

BXJ instructions are not compiler generated and are only used by java virtual machines. ARM will distribute details of this erratum to the required java virtual machine vendors.

Workaround

To prevent this erratum occurring it must be ensured that before any conditional BXJ instruction there is at least one instruction that does not set the condition codes.

For instance both of the following instruction sequences will operate correctly.

```
CMP r1,r2
LDRNE r5,[r6]
BXJNE r7
```

```
CMP r1,r2
NOP
BXJNE r7
```

 © Copyright ARM Limited 2010. All rights reserved.

324511: The contents of the IFSR may give incorrect cause of a prefetch abort.

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

The IFSR register provides the cause of an instruction abort. Under some circumstances it is possible for the IFSR to give the wrong cause of the abort.

The IFSR is updated on a prefetch abort to indicate the cause of that abort. It is a feature that was introduced in some processors prior to ARMv6, but was only formalised in the architecture in ARMv6.

When a prefetch abort is detected during an instruction fetch the cause of the abort is remembered, and if a prefetch abort is actually taken this cause of the abort is written into the IFSR.

However under the conditions described below the value written into the IFSR may not correctly reflect the abort.

Note that the term prefetch abort as used above does not include monitor mode breakpoints. If the processor takes an abort due to a monitor mode breakpoint the correct value of a debug event will be written into the IFSR.

Conditions

1. A dynamically predicted branch that is predicted as not taken.
2. A prefetch abort occurring on the dynamically predicted branch.

Implications

Most operating systems do not use the value in the IFSR to determine the value of a prefetch abort. At the time of writing this includes Linux, and other Operating Systems.

In addition, it is very unusual for a dynamically predicted branch to receive a prefetch abort. Typically this will only occur as a result of a permission failure on the branch, or a situation where the page table entry describing the location of the branch has been changed without a software invalidation of the Branch Prediction logic. For operating systems that do use the IFSR value, then if this errata is invoked the operating system will be provided with an incorrect reason for the abort. In general this erratum will cause the IFSR to contain the value 0b0000. However if the destination of the dynamically predicted branch also aborts, the IFSR will contain the abort type for that second abort. The precise implications of this erratum will vary. It is likely that either it will have no harmful side effects, or may incorrectly kill the aborting process.

Workaround

If the value in the IFSR is not used then no workaround is required. However if the value in the IFSR is used, the operating system must be tolerant to receiving an incorrect reason for the abort.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

324512: A V6 unaligned store to the first two words of a cache line in a write-through memory region can deadlock the ARM1136

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

In ARMv6, the ARM architecture introduced support for unaligned accesses to memory. This is enabled by the U bit in CP15 register 1.

A V6 unaligned store which accesses the first two words of a cache line in a write-through memory region can cause ARM1136 to deadlock if an interrupt is received whilst the unaligned store is accessing the cache. ARM1136 must be in low interrupt latency configuration.

This erratum occurs when an unaligned STR to a cacheable, write-through memory accesses the first double word in a cache line, and the STR is followed by another unaligned store or a store multiple.

The core splits the unaligned write into two accesses – one for each word. If a low latency interrupt is requested whilst the second part of the unaligned access is accessing the data cache the processor may deadlock.

Conditions

1. Low interrupt latency configuration
2. ARM1136 configured for V6 unaligned operations. i.e. CP15 Register 1, Ubit (bit 22) is set.
3. An unaligned STR to a cached write-through memory location
4. Another unaligned STR or an STM following the original STR
5. An incoming interrupt which arrives as the unaligned STR is accessing the cache.

Implications

This erratum means that unaligned accesses should not be used to a write-through memory region in low latency mode [see workarounds below]. Most operating systems use write-back memory regions as the default memory region type, so in general this erratum should not affect most application code. Application code is the most likely type of code to use unaligned accesses.

Write-through memory is mostly used only when explicitly requested by the operating system. This is likely to be for device drivers. If none of the workarounds are used, any such code must avoid the use of V6 unaligned accesses.

Workaround

If V6 unaligned accesses to write-through memory are not made then no workaround is required.

If low latency mode is not used then no workaround is required.

If low latency mode is required to ensure the precise nature of external read aborts. Then it is possible to maintain precise external aborts, but prevent this errata occurring. This involves setting the undocumented

 © Copyright ARM Limited 2010. All rights reserved.

configuration bit in Auxiliary Control Register bit [31] to 1. This bit was designed to disable the low latency interrupt mechanism which is in error, but to keep the external abort behavior of having precise aborts.

 © Copyright ARM Limited 2010. All rights reserved.

324513: PLDs may incorrectly report external aborts

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

PLD instructions are used to indicate to the memory system that memory accesses from the specified address is likely to occur in the near future. The memory system can take actions to speed up the memory accesses when they do occur. An example would be to preload the cache-line containing the address into the cache.

Due to this erratum, if an external abort occurs during the execution of the PLD instruction then that external abort will be incorrectly reported to the ARM1136 core, and the processor will take either an imprecise or a precise external abort.

Conditions

1. Use of a PLD instruction.
2. An external abort occurring when the PLD instruction attempts to preload the specified address into the cache.

Implications

The expectation is that the address accessed by the PLD instruction will be accessed in the near future, and any abort would have been reported on that later access. As a result of this erratum a PLD instruction may unexpectedly generate an abort. The implication of this erratum is that the abort will be reported earlier than originally expected. This is not expected to have any significant implications.

Workaround

It is not expected that a workaround is required for this erratum.

© Copyright ARM Limited 2010. All rights reserved.

324514: Interworking between instruction TCM and instruction Cache can result in incorrect instruction execution

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

Under rare circumstances, an instruction fetch from the instruction cache, immediately following an instruction TCM access can falsely hit in the instruction cache, and return the wrong data.

An instruction lookup to the instruction cache uses a mechanism to optimise the return of data from a linefill that is in operation due to a previous cache miss. This enables further instruction data to be returned from that same line before the linefill has been completed.

Due to this erratum, if there is an instruction TCM access followed by an instruction cache lookup, then the instruction cache lookup may return incorrect data if a previous linefill is still occurring.

An equivalent failure can occur when the TCM operates as Smartcache (so can be filled from the background memory), and a false hit is seen in the TCM.

This erratum is triggered rarely because it requires an instruction cache linefill to be taking place at the same time as a transition from instruction TCM to instruction cache lookup occurs. For this to happen the time taken to refill the last three double words of the cache line must be greater than the time taken to enter the code running in the instruction TCM, and to return to the instruction cache. Furthermore, the address of the instruction cache lookup must be to a location which, for the TCM size, would have the same memory index, but for the cache lookup do not have the same index.

Conditions

1. The instruction TCM is being used in close conjunction with the instruction cache.
2. Instruction TCM operating as local RAM, and instruction TCM is less than one quarter of the size of the instruction cache.
3. - For a 16Kb instruction cache this condition cannot be met.
4. - For a 32Kb instruction cache this implies a 4Kb instruction TCM.
5. - For a 64Kb instruction cache this implies a 8Kb or 4Kb instruction TCM.
6. Instruction TCM operating as Smartcache, and instruction TCM is not one quarter of the size of the instruction cache.

Implications

This erratum means that the use of the instruction TCM and the instruction cache in the same program with the cache sizes outlined in the Conditions above may exhibit an error in the instructions presented to the core so leading to incorrect execution.

It is not believed that the Smartcache functionality is supported by any operating systems, and that the instruction TCM, if implemented, will normally used as local RAM.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

A simple software workaround exists for the problem when the TCM is used as local RAM. This is the generally expected usage model for the TCMs. There is no workaround for this erratum if the TCM is used as Smartcache, and the TCM is not one quarter of the size of the instruction cache.

The workaround consists of making the TCM appear to be larger than it actually is. The result of this will be create multiple aliases of the TCM.

For example, if the TCM is actually 4Kbytes in size and the base address of the TCM is 0x10000, then normally the address range 0x10000 to 0x10FFF will be covered by the TCM. If the size of the TCM is forced to 8Kbytes, the effect is that the address range covered is now 0x10000 to 0x11FFF, and the addresses 0x11000 to 0x11FFF will be an alias of the address range 0x10000 to 0x10FFF.

In order to force the TCM to be larger than it actually is, the following code sequence must be used change the Cache and TCM Size configuration register:-

```
MRC p15, 0, r0, c15, c14, 0
BIC r0, r0, #&700
ORR R0, R0, #&80000000     ; this indicates a change of memory size
ORR R0, R0, #&400          ; this forces the size to 8K TCM
MCR p15, 0, r0, c15, c14, 0
```

The bit field [10:8] of this register are the only bits that should be changed in the Cache and TCM size configuration register, as these are the only bits which relate to the ITCM. Bit [31]==1 is taken to indicate a change of Cache or TCM size to the programmed value.

The encoding of the size written are:-

```
0x3 : 4KByte
0x4 : 8KByte
0x5 : 16KByte
0x6: 32KByte
0x7: 64KByte
```

All other sizes are UNPREDICTABLE.

This code sequence must be run after reset, and before the enabling of the TCM.

 © Copyright ARM Limited 2010. All rights reserved.

325157: Associative ICache maintenance operations can deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

Under very rare circumstances, an associative invalidation of an instruction cache line, or a pre-fetch of an instruction cache line, can deadlock the system.

The interaction between CP15 instruction cache operations which access the instruction TLB and the normal pre-fetching from the instruction cache has an errata whereby the CP15 instruction main TLB lookup can be lost due to a second main TLB lookup being required from the normal instruction pre-fetching. This results in a system deadlock.

This errata occurs if, while a CP15 generated instruction TLB miss is outstanding, the instruction pre-fetching causes a microTLB miss, hence requiring a main TLB lookup, on a speculative instruction fetch which is subsequently cancelled.

This event occurs under a very rare set of circumstances, requiring two branch predicted taken branches to occur ‘back to back’ within 2 instructions of the cache maintenance operation, and the first of these branches must miss in the instruction cache (or TCM). The second branch must lie on a page boundary, such that the subsequent instruction is a new page (which causes a micro-TLB miss)

Conditions

1. Branch prediction is enabled.
2. MCR p15, 0, c7, c5, 1 (invalidate instruction cache line by MVA) or MCR p15, 0, c7, c13, 1 (pre-fetch instruction cache line by MVA).
3. A branch as the subsequent instruction or as the next following instruction.
4. The target of that branch being itself a branch, at the final address of a page.

Implications

The combination of conditions to the instruction fetch is unlikely to occur in most code, as the cache maintenance operations are rarely used, and are expected to be contained within HAL code, and require the processor to be in a privileged mode to be execute. This makes analysing code for such problems more straightforward, as is applying workarounds.

Where the critical code sequence does arise, the result is a deadlock of the core.

Workaround

In the rare cases that a workaround is needed, a simple workaround of ensuring that a branch does not occur within 2 instructions of the instruction cache maintenance operation can be applied, which will remove any danger of the problem arising.

 © Copyright ARM Limited 2010. All rights reserved.

326103: FSR write bit incorrect on a SWP to read-only memory

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

The FSR bit[11] value indicates, after an abort, if the instruction was performing a read or a write. A typical use of this is to detect writes to read-only memory, to implement a copy-on-write scheme. Because of this errata the FSR bit indicates a read on a SWP that aborts because of a TLB generated abort (even though the abort arose as a result of a write to read-only memory).

A SWP operation performs a read followed by a write. The permission check in the TLB for the read is designed to check that both the read and the write will pass their check, to avoid issuing the read when the write aborts. This avoids problems on the AHB interface where the read part of the locked transfer occurs without the following write part. As a result of this behaviour, perfectly correctly, the read will abort if the memory is not writeable. The aborting read then prevents the write part from occurring.

This erratum causes the read part of the SWP to be marked as a read in the FSR when the abort is taken, even if the abort was due to not being able to write the memory.

Conditions

1. A SWP or SWPB to a read-only region of memory

Implications

The Write Bit of the FSR was introduced to allow operating systems to be able to use copy-on-write approaches without having to examine the instruction to discover the source of the permission failure. Copy-on-write is used to allow processes to share data structures, and to take a copy of the data structure only when a write access is performed to it. Typically this is implemented by marking the region as read-only, and faulting on the process attempting to write the data.

On Linux, the effect of this erratum is that the abort does not trigger the copy-on-write routine, because the access is not recognised as a write. Instead the aborting code is restarted, and aborts again, so livelocking the operating system.

On other operating systems that make use of the FSR write bit to implement copy on write, an unexpected abort could be flagged by the core, resulting in an unexpected error, which is likely to be terminal to the process.

Workaround

This erratum can be worked around in the abort handler of the operating system by adding code to check if the aborting instruction is a SWP, and correcting the FSR Write bit appropriately. In general, such code can be added in a way that does not significantly compromise the performance of the abort handler.

A typical piece of code of this form would be:

```
MRS     r3, SPSR                ; r3 contains the SPSR
TST     r3, #0x20               ; check T bit
```

 © Copyright ARM Limited 2010. All rights reserved.

```
TSTEQ  r3, #0x01000000          ; check J bit
LDREQ  r3, [r14, #8]            ; get instruction
ANDEQ  r0, r3, #0x0ff00000      ; Check bits 20-27
CMPEQ  r0, #0x01000000          ;
ANDEQ  r0, r3, #0xf0            ; Check bits 4-7
CMPEQ  r0, #0x90                ;
MRC    p15, 0, r1, c5, c0, 0    ; get FSR
MRC    p15, 0, r0, c6, c0, 0    ; get FAR
ORREQ  r1, r1, #1<<11           ; set write bit if !T, !J, SWP
```

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

326295: EDBGRQ does not cause Debug halting when Debug is not enabled

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

The assertion of the EDBRQ pin, or the sending of a Halt command on JTAG, does not have an effect on the core if Halt-mode debug is not enabled. [e.g. when enabled for monitor mode].

EDBGRQ was specified as part ARMv6 Debug not to have an effect if Halt-mode debug was not enabled. In this, the behaviour was different from that of previous ARM cores, and the decision has been taken to revert to the original behaviour on future cores implementing ARMv6 debug.

The same change of specification has been agreed to apply to debug entry triggered by JTAG using the Halt instruction in the JTAG TAP controller.

This erratum is a specification change erratum.

Conditions

DSCR[14] = 0 (Debug halt mode disabled)

and either

1. EDBGRQ used to halt the core
2. or a HALT instruction is sent using the JTAG TAP controller

Implications

This erratum means that debug must be enabled to allow the EDBGRQ (or the JTAG Halt command) to halt the core.

Workaround

Systems that require EDBGRQ (or the JTAG Halt command) to halt the core must set DSCR[14].

 © Copyright ARM Limited 2010. All rights reserved.

326296: Potential incorrect operation of Clean Entire / Clean and Invalidate Entire Data Cache operations in low latency mode

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

In Low Interrupt Latency configuration, under rare circumstances, an interrupt arriving during a Clean Entire Data Cache or a Clean and Invalidate Entire Data Cache operation can result in incorrect behaviour.

The Clean Entire Data Cache and the Clean and Invalidate Entire Data Cache operations have an optimisation that if the operation is interrupted, and the interrupt service routine does not cause the cache to be dirtied (that is, does not cause a write to writeback memory that hits in the cache), then the operation is restarted from the point that it was interrupted, rather than from the start of the operation.

Due to this erratum there are two scenarios in Low Interrupt Latency configuration which can cause incorrect operation related to these instructions:

1. If an interrupt arrives during the first cycle of a Clean Entire Data Cache or a Clean & Invalidate Entire Data Cache operation that has been interrupted previously, then all subsequent loads and stores might have their addresses generated incorrectly.
2. A flag in the Cache Dirty Status Register (MCR p15, 0, Rx, C7, C10, 6) indicates when the Data Cache is dirty and hence the clean operation needs to be restarted from the beginning. If an interrupt arrives at the end of a Clean Entire Data Cache or a Clean and Invalidate Entire Data Cache operation, then the Cache Dirty Status Register might get incorrectly cleared. As a result the Data Cache will be marked clean even though the Data Cache might still contain dirty lines.

Conditions

1. The core is configured with Low Interrupt latency configuration.
2. A 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) is performed with interrupts enabled.
3. The interrupt service routine does not perform a write to Writeback memory that hits in the cache.

Implications

The effect of this erratum depends on when an interrupt arrives:

1. If an interrupt arrives during the first cycle of the Clean (and Invalidate) Entire Data Cache operation, the effect is fatal to the correct operation of the processor, as all subsequent data accesses are corrupted.
2. If an interrupt arrives at the end of the Clean (and Invalidate) Entire Data Cache operation, the effect is that the Cache Dirty Register is incorrectly cleared. As a result the Data Cache will be marked as clean even though the Data Cache might still contain dirty lines.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

1. A workaround can be invoked which increases the worst case interrupt latency to that which exists when the low interrupt configuration is disabled, but keeps the precise nature of external read aborts. This workaround involves setting the undocumented configuration bit in Auxiliary Control Register bit [31] to 1. This bit was designed to disable the mechanism which is in error, but to keep the external abort behaviour of having precise aborts.
2. All code that executes the clean entire data cache or clean and invalidate entire data cache commands is run with interrupts disabled.
3. The use of the Clean (and Invalidate) Entire Data Cache operation can be replaced by a software loop that uses the set/index representation of the cache lines to clean (and invalidate) each data cache line.
4. The use of Clean (and Invalidate) Entire Data Cache operation can be configured to cause an UNDEFINED instruction trap by the setting of an undocumented bit, bit [4] in the Auxiliary control register. This bit has no other effect. The Undefined instruction handler can then be configured to insert a software loop which has the same effect on the data cache.

 © Copyright ARM Limited 2010. All rights reserved.

328395: In Low Interrupt Latency configuration, an interrupt to a SWP on to non-cachable memory can result in a system deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

If a SWP instruction is executed to a normal noncachable (or a normal sharable) location with address bits [4:3] = 0b11 when in low interrupt latency configuration, the interrupt might be recognised by the core between the load part of the SWP and the store part of the SWP. This results in the load part of the SWP being performed but the store part of the SWP not being performed on the bus interface. This means that the HMASTLOCKR signal remains asserted from the Load part of the SWP until a subsequent memory transaction is performed on the data read port. This continuous assertion of the HMASTLOCKR signal might lead to problems in the system.

Conditions

1. Low Interrupt latency configuration is in use
2. A SWP is made to an normal non-cachable or a normal sharable location
3. The SWP is to an address with bits [4:3] = 0b11
4. Interrupts are enabled during the SWP

Implications

The majority of operating systems currently implemented make very little use or no use of noncachable normal memory or sharable normal memory. Noncachable normal memory and sharable normal memory are new memory types for ARMv6. In addition, the use of such regions will be limited to specific drivers which need to have these memory characteristics.

When this problem arises, the system can become deadlocked because the data write port, the peripheral port and the instruction port cannot be serviced until a further read from the data read port is performed. For example, if the instruction fetches for the interrupt service routine make a cache miss to the same memory slave as the SWP, the instruction fetch cannot be serviced, and this deadlocks the system.

Workaround

There are three possible workarounds that can be applied if necessary:

1. If interrupts are disabled for the duration of the SWP, this problem is avoided. Disabling interrupts for such SWPs is a practical solution because there are not likely to be many cases where interrupts are required.
2. Running the core in low interrupt latency configuration with the undocumented Auxiliary control bit[31] set avoids this problem, because it prevents the interrupt from being recognised between the load and store.
3. If the region of memory that the SWP is being performed to is marked as Strongly-Ordered or marked as Device, the problem does not arise, and the location is not cached.

 © Copyright ARM Limited 2010. All rights reserved.

328409: Externally aborted loads to the PC can result in deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

If a load to the PC, including LDM to the PC or RFE, receives an external abort while there is an outstanding load to the peripheral port, the load from the peripheral port is incorrectly discarded, and the result of the load is lost. This leads to the core failing to clear the outstanding transaction from the core scoreboard. This can result in a core deadlock, because any following instruction that depends on the result of the load from the peripheral port stalls indefinitely.

In practice, this problem only occurs if the peripheral port access takes at least two core cycles longer than the time taken to return an aborted read on the data read port

Conditions

1. Core not in Low Interrupt Latency configuration
2. A load from the peripheral port has been issued, but the result is not yet returned
3. A load to the PC or an RFE receives an external abort

Implications

This failure can be seen in systems which generate external aborts from the Data Read AHB port, and also make use of the Peripheral AHB Port for read accesses, where the read access latency is greater than two cycles longer than the minimum latency on the Data Read port.

This failure leads to a core deadlock around the time that an external abort has been reported to the core. Such an external abort, which has led to a corruption of the program counter, is likely to cause a fatal abort at least to that process and in many systems to the entire system. However, the effect of deadlocking the core prevents a software triggered reset being applied, and instead the system is likely to be reset using a system watchdog if one is present, or to deadlock indefinitely otherwise.

It is usual that the peripheral port is accessed by specific device code, and this provides more scope for the application of a workaround.

Workaround

Two possible workarounds exist to this erratum:

1. Insert a DrainWriteBuffer instruction between every read to the peripheral port and a subsequent load to the PC that occurs before the use of the returned data. Generally, Peripheral port accesses are contained within specific HAL layer driver code (where interrupts can be disabled to avoid the arrival of an interrupt between the peripheral port load and the DrainWriteBuffer).
2. Where the guarding of reads from the peripheral port cannot be implemented as in workaround 1, the use of the low interrupt latency configuration, combined with the undocumented Auxiliary Control

 © Copyright ARM Limited 2010. All rights reserved.

Register bit [31], avoids this problem. This control bit, combined with low interrupt latency configuration, ensures the in-order return of all data without also reducing the interrupt latency.

 © Copyright ARM Limited 2010. All rights reserved.
Non Confidential

328428: External abort on the non requested word of a cache linefill can lead to cache corruption

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

The signalling of an external abort on a doubleword of a cache linefill that was not the doubleword that caused the linefill to occur causes the linefill to be abandoned, but does not cause an external abort to be signalled to the core. If, between the start of the linefill and the signalling of the external abort, a write is made to a word which has been returned to the cache in the cache line that is being filled, then this write updates both the cache and the external memory. Because of this erratum, the write to the cache line may, under very rare circumstances, be applied to the same physical location (that is the same Set/Way in the data cache) after that entry has been filled by a subsequent cache linefill of a different address. This leads to a corruption of a word cached as a result of the subsequent linefill.

Conditions

1. External aborts can be applied to individual doublewords of a cache line fill rather than applying to all entries in a cache linefill.
2. A write to the currently refilling line during a linefill
3. A subsequent filling of that same location in the cache by a different address

Implications

The use of external aborts is typically split into coarse granularity use, where all doublewords in a cache line are aborted during their fetch from memory if there is an external abort, and fine granularity use, where a single doubleword in the sequential accesses of the linefill can generate an abort. Coarse granularity use is common where external aborts are used for protection checking. Fine granularity use is more common with systems where the integrity of the memory is checked on each lookup, for example using parity. This erratum can only arise with the fine granularity use of external aborts.

On the rare occasions that this erratum arises, typically after a parity error during a cache linefill, the effect is to corrupt data in the cache, resulting in a corruption of memory and subsequent incorrect operation.

In the case of the ARM L210 Cache Controller, all parity errors generated from within the L210 Cache controller are applied to all words of the linefill, so this erratum is not caused by parity errors in the L210 Cache.

Workaround

The only workaround for this problem is that external aborts must not be used in a manner where the granularity of the abort is within a cache line. This enables external aborts to be used for the protection of regions, for example for security, but limits their use for parity or ECC checking.

Where parity or ECC checking is used, the failure of the check should be signalled by the use of interrupts, rather than the use of external aborts.

 © Copyright ARM Limited 2010. All rights reserved.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

328429: An Associative Invalidate Instruction Cache or Prefetch operation can result in microTLB corruption

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

If an instruction cache invalidate by MVA (including instruction invalidate range) or Prefetch by MVA operation that missed in the instruction micro-TLB is interrupted, and the fetch from the interrupt vector causes an instruction micro-TLB miss, this erratum results in the instruction micro-TLB being corrupted. The corruption causes the virtual address of the interrupt vector micro-TLB entry to be associated with the physical address and attributes of the address passed in the instruction cache maintenance operation. As a result, the physical addresses which are used for the interrupt handler instruction fetches are incorrect, giving incorrect execution. This erratum arises only in a one cycle timing window for the arrival of the interrupt, and so only occurs very rarely.

As a related manifestation of this erratum, the use of the background prefetch instruction cache range operation can cause a corruption of the instruction micro-TLB without the intervention of an interrupt when an instruction micro-TLB miss coincides with the return of a micro-TLB miss from the background prefetch instruction cache range operation. This means that the use of the background instruction cache prefetch range operation is unsafe.

Conditions

1. An instruction cache invalidate by MVA or an instruction cache invalidate range or an instruction cache prefetch by MVA is run with interrupts enabled, OR
2. An instruction cache prefetch range is run

Implications

The corrupted micro-TLB entry causes the instruction fetch of the vector location to come from the wrong physical address, so leading to incorrect operation, so the consequences of this problem are significant. The consequences vary for the different instruction cache maintenance operations in use:

1. Typically the prefetch instruction cache line by MVA operation is performed as part of a lockdown sequence, and as such is performed with interrupts disabled, so avoiding a problem from this operation.
2. The invalidate instruction cache line by MVA operation is often called from within self-modifying code sequences (such as within Java handling) and other key situations such as when code is loaded. The call to this operation is embedded within function calls in the operating systems, and those functions can be customised as part of the CPU specific code. This allows one of the workarounds to be applied as part of the CPU specific code.
3. The invalidate instruction cache range by MVA operation is not commonly used within operating systems, but where is used, it is embedded within CPU specific functionality.
4. The prefetch instruction cache range operations are not commonly used, but are available to user code, so the workaround to disable range operations should be applied.

 © Copyright ARM Limited 2010. All rights reserved.

Note that this erratum has similar implications to erratum 371025, which is not fixed in r1p0.

Workaround

The prefetch instruction cache prefetch range operation must not be used. If there is a danger of user code using this functionality, the operation can be prevented by setting the Auxiliary Control Register RV bit (bit[5]). This causes an undefined instruction trap if they are used. This will also prevent the use of the instruction cache invalidate range operation.

The prefetch instruction cache line operation must be used only when interrupts are disabled. This is likely to be the case for the common usage model of this operation (for instruction cache lockdown).

The problem can be worked around for the invalidate instruction cache by MVA operation by:

1. Executing this operation only when interrupts are disabled or
2. Replacing this operation by the use of four invalidate by Set/Way operations with interrupts disabled as described below or
3. Replacing this operation by an Invalidate Entire Instruction Cache operation. You must use the workaround documented in erratum 411920 to perform this operation safely.

The invalidate instruction cache by MVA operation can be replaced by four invalidate by Set/Way operations, where the Set number takes the virtual index, and the way value is cycled between 0 and 3. Note that if the SmartCache is used, then the Set/Way operation must be performed on the SmartCache as well.

Interrupts must be disabled during this process because of erratum 405875.

Thus:

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

can be replaced by:

```
MOV    Rtmp1, #0
MRS    Rtmp2, cpsr                ; save cpsr
CPSID  ifa                        ; disable interrupts
ORR    Rtmp1, rx, #0xC0000000
BIC    Rtmp1, Rtmp1, #1
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way3
SUB    Rtmp1, Rtmp1, #0x40000000
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way2
SUB    Rtmp1, Rtmp1, #0x40000000
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way1
SUB    Rtmp1, Rtmp1, #0x40000000
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way0
[ORR   Rtmp1, Rtmp1, #1
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate SmartCache]
MSR    cpsr_cx, Rtmp2             ; restore interrupts
```

 © Copyright ARM Limited 2010. All rights reserved.

328430: A precise external data abort followed immediately by a TLB generated data abort can result in an incorrect DFSR and FAR

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

On the cycle that a precise external abort is acknowledged by the core, the DFSR and FAR values are written. As a result of this erratum, if in that same cycle, a TLB generated abort on a subsequent transaction (which will be discarded due to the precise external abort) occurs, this can result in the contents of the DFSR and FAR being reset to 0.

This erratum occurs only rarely. It results from an unusual timing of the instructions in the core pipeline.

Conditions

1. Not using Low Interrupt Latency configuration
2. A precise external data abort is generated
3. In the same cycle a TLB generated data abort is generated for a subsequent transaction

Implications

This erratum results in the corruption of the FAR and DFSR in the case of an external abort under rare circumstances. The resulting corrupt DFSR is a RESERVED abort encoding, and so is expected to be handled in the operating system as an unexpected abort, resulting in a fatal error. For most operating systems, the return of an external abort is seen as a fatal event in any case, so the result of this is a badly reported fatal error, but is not expected to result in significant problems.

Workaround

In many cases, no workaround is needed for this problem. Where a workaround is required, two approaches can be used:

1. Running the core in Low Interrupt Latency configuration with the undocumented Auxiliary control bit[31] set avoids this problem, because it ensures in-order return of all data.
2. Installing a handler for the DFSR[10,3:0] = 0x0 case to distinguish the encoding as describing an external abort enables an operating system to distinguish this case uniquely, so allowing more accurate reporting of the problem. However, the Fault Address Register is also lost, so the handler must examine the failing instruction to determine the faulting address in this case, if precise recovery is to be achieved.

 © Copyright ARM Limited 2010. All rights reserved.

329839: In low interrupt latency mode a TLB generated abort of a V6 unaligned load or store can result in a core deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

When the core is configured to run with low interrupt latency configuration, and v6 unaligned handling is enabled, then a TLB generated abort (including an alignment fault) on an unaligned LDR or LDRH can result in a core deadlock situation if the first load or store executed from the abort exception is an LDM or STM.

Conditions

1. Running in low interrupt latency configuration
2. V6 unaligned handling is enabled
3. A TLB generated abort occurs on an unaligned transaction
4. The first load or store executed from the exception entry is a LDM or STM.

Implications

This problem is unlikely to be seen in many practical applications because most ARM operating systems use an LDR to the PC as the first instruction of the exception handler - this load is placed at the exception vector, and avoids the problem. This is true for Linux, and other Operating Systems.

In addition, many ARM operating systems set the CP15 unaligned trap and do not expect unaligned code to exist at present, which also reduces the likelihood of seeing this problem.

Workaround

Setting of the undocumented CP15 Auxiliary Control bit[31] which keeps the precise data abort functionality for low interrupt latency operation, but increases the worst case interrupt latency, will work around this problem.

© Copyright ARM Limited 2010. All rights reserved.

333030: The GE flags might not be updated

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

Under rare conditions, a SIMD instruction might not update the GE flags (bits[19:16] of the(CPSR).

In ARMv6, the SIMD instructions use CPSR bits[19:16] as Greater than or Equal (GE) flags for individual bytes or halfwords of the result. These flags can be used to control a later SEL instruction. The following SIMD instructions write the GE flags; SADD16, SSUB16, SADDSUBX, SSUBADDX, SADD8, SSUB8, UADD16, USUB16, UADDSUBX, USUBADDX, UADD8, USUB8.

A MCR to PC instruction specifies the program counter as the destination register (<Rd>), an example is the FMSTAT floating point instruction. When executed this instruction will write only the N, Z, C and V flags in the CPSR.

If a SIMD instruction which sets the GE flags is directly followed by a MRC to PC instruction, then write to the GE flags might be masked. As a result, the GE flags will not be updated.

Conditions

1. An instruction which stalls in the Wr pipeline stage.
2. Directly following this instruction is a SIMD instruction which sets the GE flags.
3. The SIMD instruction is either: a) Directly followed by a MRC to PC instructions, or b) Directly followed by a branch to a MCR to PC instructions. In the second case the branch must be folded onto the MCR to PC instruction.

Implications

Under the conditions described above the SIMD instruction will fail to set the GE flags.

Unless a coprocessor (including the VFP) is being used this erratum will have no effect.

It is very unlikely that the SIMD instructions will be mixed with a floating point or coprocessor instructions.

Currently the ARM compiler does not generate the SIMD instructions which set the GE flags.

Workaround

To prevent this erratum occurring it must be ensured that before any MRC to PC instruction there is at least one instruction that does not set the GE flags.

For example, the following instruction sequence will operate correctly.

```
UADD8 r2, r3, r4
NOP
FMSTAT
```

 © Copyright ARM Limited 2010. All rights reserved.

334358: Mispredicted branch folded onto an MCR can be missed from ETM trace

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

When executing two instructions that access different coprocessors, the second instruction may be stalled until the completion of the first. Should a mispredicted branch instruction be folded onto the second coprocessor instruction in this scenario, the branch will be omitted from the ETM trace.

Conditions

1. The coprocessor instruction accesses a different coprocessor to the previous coprocessor access.
2. A branch is folded onto the coprocessor instruction.
3. The branch is mispredicted.

Implications

Instruction trace will become invalid until the next indirect branch is seen, or I-sync is output. This should not affect data trace. Address comparators based on instruction addresses will not be impacted by this unless sensitive to the missed instruction.

Workaround

There is no known workaround for this erratum. Faster recovery of trace from this scenario can be achieved by increasing the frequency of periodic I-sync packets output by the ETM.

© Copyright ARM Limited 2010. All rights reserved.

334360: A coprocessor instruction that stalls and then is cancelled by an interrupt loses ETM trace sync

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

VFP and GCP instructions may stall the processor prior to execution. If an interrupt is recognised in this period, and the VFP/GCP instruction cancelled so the interrupt can be processed, then the ETM can lose synchronisation.

Conditions

1. A VFP or GCP instruction is executed
2. The VFP/GCP instruction is cancelled by an interrupt whilst it is stalled

Implications

Data trace may become invalid until both the ETM and Core are reset. Address comparators based on data trace will behave unpredictably. This will not affect instruction trace unless the user has set up trace regions based on data comparisons. In this case, although instruction trace will be valid, it may not reflect the intended trace region.

Workaround

There is no known workaround for this erratum.

 © Copyright ARM Limited 2010. All rights reserved.

335567: STREX{B|H|D} instructions might return incorrect status

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

STREX (Store Register Exclusive) performs a conditional store to memory. The store only occurs if the executing processor has exclusive access to the memory addressed. This operation returns a status value. If the store updates memory the return value is 0, otherwise it is 1.

STREX is used in combination with LDREX to implement inter-process communication in multiprocessor and shared memory systems.

Under rare conditions, it is possible that STREX to memory regions that have the Shared attribute will return incorrect status value. Instead of operation failure (status value 1), the returned status might indicate that the operation has updated memory.

Conditions

1. A LDREX to a memory region that has the Shared attribute.
2. A STREX to a memory region that has the Shared attribute. The status value is being returned to register Rd.
3. Optionally, any number of instructions that does not use register Rd.
4. A LDREX that does not use register Rd.
5. A STREX that does not use register Rd.

Implications

Under conditions described above, this erratum might result in an incorrect status value being returned for the first STREX instruction. Instead of indicating that the STREX operation has failed, the status value might incorrectly indicate that the operation has succeeded.

Expected use of LDREX and STREX instructions is to execute LDREX, followed by STREX and then examine the returned status value. Given below is an example of typical usage:

```
    Lock address :LockAddr
    Lock free    : 0x00
    Lock taken   : 0xFF

    MOV     R1, #0xFF            ;load the 'lock taken' value
try LDREX   R0, [LockAddr]       ;load the lock value
    CMP     R0, #0               ;is the lock free?
    STREXEQ R0, R1, [LockAddr]   ;try and claim the lock
    CMPEQ   R0, #0               ; did this succeed?
    BNE     try                  ;no - try again. . . .
                                 ;yes - we have the lock
```

 © Copyright ARM Limited 2010. All rights reserved.

In this code example, STREX returns the status value to R0 and the following instruction (CMPEQ) uses the same register. This will ensure that the conditions for the errata are not satisfied.

Also, it is architecturally required that a dummy STREX is executed on a context switch or exception. This will ensure that the conditions for the errata are not satisfied.

Workaround

STREX instruction should be followed by an instruction that uses the returned status. For example;

```
STREX r0, r1, [r2]
MOV r0, r0
```

 © Copyright ARM Limited 2010. All rights reserved.

336501: Wait For Interrupt does not drain the Data Cache write buffer

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

Dormant mode enables the core to be powered down, leaving the caches and the Tightly-Coupled Memories (TCM) powered up and maintaining their state.

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. When a WFI operation is executed, the STANDBYWFI output signal is asserted to indicate that the processor is in Standby mode and that Dormant mode can be entered.

The WFI operation does not drain the Data Cache write buffer before the STANDBYWFI output signal is asserted. If the Data Cache write buffer contains valid entries and the Dormant mode is entered, all valid entries in the buffer will be lost. This will result in the Data Cache getting corrupted.

Conditions

1. A store instruction that hits the Data Cache.
2. Optionally, any instruction that does not drain the Data Cache write buffer (*).
3. A WFI operation followed by a Dormant mode entry.

(*) Only Clean and/or Invalidate Data Cache Maintenance operations are guaranteed to drain the Data Cache write buffer.

Implications

Under the conditions described above, the Data Cache will get corrupted.

Workaround

To prevent this erratum occurring it must be ensured that before the WFI operation is executed, a Clean Data Cache Line operation is performed. For example:

```
MOV r0, #0
MCR p15, 0, r0, c7, c10, 2  ; Clean Data Cache Line (using index)
MCR p15, 0, r0, c7, c10, 5  ; Data Memory Barrier
MCR p15, 0, r0, c7, c0, 4   ; WFI
```

Note that the cache operation is required only for Dormant mode entry, and not if the WFI is used only to enter Standby mode.

 © Copyright ARM Limited 2010. All rights reserved.

336509: Disabling the loading of the main TLB does not prevent updates to the lockdown region

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

The loading of the main TLB after a hardware page table walk can be disabled by setting the IML and DML bits in the CP15 TLB Debug Control register. If the loading of the main TLB is disabled, then misses do not result in the main TLB being updated. This has a significant impact on performance, but enables debug operations to be performed as unobtrusively as possible.

Hardware page table walks can place the TLB entry in the set associative region or the lockdown region of the main TLB. The CP15 TLB Lockdown register controls which region is used and which entry is written in the lockdown region. If the P bit in the TLB Lockdown register is set, then hardware page table walks will place the TLB entry in the lockdown region at the entry specified by the victim field of the TLB Lockdown register. The victim automatically increments after any page table walk that results in an entry being written into the lockdown region of the TLB.

If the P bit in the TLB Lockdown register is set and the loading of the main TLB is disabled, then page table walks should not result in the main TLB being updated. However, due to this erratum, page table walks will place the TLB entry in the lockdown region and increment the victim.

Conditions

1. The IML and DML bits in the TLB Debug Control register are set.
2. The P bit in the TLB Lockdown register is set.
3. A hardware page table walk is performed, and the page table walk returns a valid TLB entry.

Implications

As a result of this erratum, the main TLB state might unexpectedly be changed. More precisely, the lockdown region of the main TLB might be polluted with unexpected TLB entries and the lockdown victim counter incremented to an unexpected value.

Practically, this erratum is expected to have consequences only when debugging code.

Workaround

On entry to the Debug state, the debugger needs to save the TLB Lockdown register and clear the P bit of this register before making any memory access.

Immediately before exit from the Debug state, the debugger has to restore the TLB Lockdown register.

 © Copyright ARM Limited 2010. All rights reserved.

343942: VFP load/store multiple instructions of length > 16 words that cross memory regions can result in a deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

VFP or GCP instructions can issue multiple access data transfers of length greater than 16 words using, for example, LDC, STC, FLDMD or FSTMD instructions. These long transfers can potentially cross a boundary between two memory regions. The second memory region can abort for multiple reasons (permission fault, Access Bit Fault if ForceAP is enabled,...). If such an abort occurs, the core can deadlock.

Conditions

1. System is running with Low Interrupt Latency Configuration disabled
2. A VFP or GCP instruction launches a multiple access data transfer of more than 16 words
3. The start address of the transfer is 16 words below the end address of the first memory region
4. The transfer crosses a boundary between two memory regions
5. The second memory region aborts

Implications

Under the conditions described above, a deadlock can result.

Note that code compiled with version 2.2 of the RealView Developer Suite Compilation Tool (RVCT) will not exhibit this erratum.

Workaround

To ensure that this erratum is avoided, configure the design to function in the low interrupt latency configuration and set the undocumented Auxiliary Control Register bit [31]. This control bit was designed to disable the mechanism which is in error, but to keep the external abort behaviour of having precise aborts.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

344158: Externally aborted LDM to the PC might not take data abort exception

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

If a multiple word load including the PC (LDM to the PC) receives an external abort, the LSU will not pass the abort to the core. This leads to the LDM instruction retiring without taking the data abort exception.

An optimisation in the ARM1136 causes the load of the PC to be executed first, followed by the load of the remaining registers in the original specified order. If the load to PC receives an external abort, the LSU will incorrectly acknowledge the LDM instruction without passing the abort to the core. As a result, the LDM instruction will retire without taking the data abort exception. Instead, the data abort will be either lost or taken on the next load/store instruction.

Conditions

1. System is running with Low Interrupt Latency Configuration disabled.
2. An LDM of at least two registers (including the PC).
3. The load to PC receives an external abort.

Implications

This erratum can be seen in systems which generate external aborts from the Data Read AHB port.

Under the conditions described above, this erratum will result in either the external abort not being taken at all, or the external abort being taken on a load/store instruction following the LDM to the PC. This can result in incorrect program execution.

Workaround

To ensure that this erratum is avoided, configure the design to function in the low interrupt latency configuration and set the undocumented Auxiliary Control Register bit [31]. This control bit was designed to disable the mechanism which is in error, but to keep the external abort behaviour of having precise aborts.

 © Copyright ARM Limited 2010. All rights reserved.

349888: Enabling the Cache Size Restriction can result in a deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

The CZ bit, bit 6 in the CP15 Auxiliary Control Register, controls the restriction of cache size to 16 kB. When the CZ bit is set, the size of the Data and Instruction Caches is limited to 16 kB. This enables the processor to run software that does not support ARM v6 restrictions on page table mappings, even if the processor is implemented with caches larger than 16 kB.

If the Instruction Cache size is less than 16 kB and the CZ bit is set, this can result in incorrect program execution or even a deadlock can occur

Conditions

1. The Instruction Cache size is 4 or 8 kB.
2. The CZ bit is set.

Implications

This erratum means that operating systems, which do not implement ARM v6 restrictions on page table mappings, can not be run on processors with Data Cache size of 64 or 32 kB and the Instruction Cache size of 8 or 4 kB.

This erratum can result in a deadlock.

Workaround

Workaround for this erratum is set the CZ bit to one only if the instruction cache size is 16 kB or more.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

351914: The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (MOV PC)

Status

Affects: product ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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

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.

Implications

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 erratum will be seen in any real systems.

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 operation can be tolerated, for example, a system using floating point for graphics, no workaround should be needed. In other systems, where the very

 © Copyright ARM Limited 2010. All rights reserved.

occasional error is not acceptable, the workaround must be employed. However, we do not expect that this code sequence will be used.

Workaround

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

See also the related erratum:

351912: The VFP11 double-precision multiply or multiply-accumulate operation can corrupt data (branch folding)

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

364296: Possible Cache Data Corruption with Hit-Under-Miss

Status

Affects: product ARM1136J-S, ARM1136JF-S.

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

Description

When working with Hit-Under-Miss enabled, a very rare combination of circumstances can lead to corruption in the data cache.

Data being written into the cache is buffered in a buffer before the data is written to the cache. If the line to which this data is being written is evicted from the cache in the cycle after the write has been committed to the buffer, the buffer entry is not invalidated. As a result, the line in the buffer can, under rare circumstances, be written incorrectly to the new line that has filled into the cache in place of the evicted line. This leads to data corruption.

The corruption in the cache applies to both Write Back and Write-Through memory.

Conditions

1. Core not configured to run in Low-Interrupt Latency configuration
2. Writes to cacheable memory (either write-back and write-through)

Implications

The corruption in the data cache leads to incorrect operation. The workaround outlined below must be applied.

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 mode, 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.

© Copyright ARM Limited 2010. All rights reserved.

371025: Associative Invalidate Instruction Cache operation can fail

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

Under very rare conditions, an associative Invalidate Instruction Cache operation can fail to invalidate a cache line or can mark other cache lines as valid.

The following two CP15 operations are affected:

1. Invalidate Instruction Cache using MVA (MCR p15, 0, <rd>, c7, c5, 1)
2. Invalidate Instruction Cache range (MCRR p15, 0, <end_addr>, <start_addr>, c5)

Conditions

Condition set 1

1. An associative Invalidate Instruction Cache operation is executed.
2. The operation misses the microTLB and hits the Instruction Cache.
3. Immediately after the CP15 microTLB miss, the Prefetch Unit performs a normal instruction prefetch which misses the microTLB.

Condition set 2

1. An associative Invalidate Instruction Cache operation is executed.
2. The operation hits the microTLB and hits the instruction TCM.
3. An interrupt occurs.

It should be noted that particular timings of events are required in addition to the above sets of 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 three workarounds can be applied for this erratum.

Workaround 1

The Invalidate Instruction Cache using MVA operation can be replaced by four Invalidate by Set/Way operations, where the Set number takes the virtual index, and the way value is cycled between 0 and 3. Note that if the SmartCache is used, then the Set/Way operation must be performed on the SmartCache as well.

 © Copyright ARM Limited 2010. All rights reserved.

Thus:

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

can be replaced by:

```
MOV    Rtmp1, #0
MRS    Rtmp2, cpsr                ; save cpsr
CPSID  ifa                        ; disable interrupts
MCR    p15, 0, Rtmp1, c7, c12, 5  ; Stop Prefetch Range
ORR    Rtmp1, rx, #0xC0000000
BIC    Rtmp1, Rtmp1, #1
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way3
SUB    Rtmp1, Rtmp1, #0x40000000
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way2
SUB    Rtmp1, Rtmp1, #0x40000000
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way1
SUB    Rtmp1, Rtmp1, #0x40000000
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate way0
[ORR   Rtmp1, Rtmp1, #1
MCR    p15, 0, Rtmp1, c7, c5, 2   ; invalidate SmartCache]
MSR    cpsr_cx, Rtmp2             ; restore interrupts
```

To replace the Invalidate Instruction Cache Range operation, use the above code in a loop.

This workaround is very similar to workaround 2 documented in erratum 328429. The workaround specified above also protects against background prefetch operations, which is not necessary in erratum 328429 because background instruction cache prefetch range operations are unsafe in revisions of ARM1136 affected by erratum 328429.

Note that the Stop Prefetch Range instruction results in an Undefined Instruction Exception if:

1. The Auxiliary Control Register RV bit (bit[5]) is set.
2. The instruction is executed on other processors in the ARM11 family.

If you need to implement a single workaround that can be executed on other processors in the ARM11 family, then you must remove the Stop Prefetch Range instruction from the workaround and ensure that the Auxiliary Control Register RV bit (bit[5]) is set. This will disable all Prefetch Range instructions, causing them to generate an Undefined instruction exception.

Workaround 2

Invalidate the microTLB immediately before the invalidate operation. This workaround requires that the uTLB random replacement must be disabled (bit[3] in the CP15 Auxiliary Control register must be zero). Replace:

```
MCR  p15, 0, rx, c7, c5, 1
[MCRR   p15, 0, ry, rx, c5]
```

with:

```
MRS    Rtmp1, cpsr                ; save the CPSR
CPSID  ifa                        ; disable interrupts
MOV    Rtmp2, #0
MCR    p15, 0, Rtmp2, c13, c0, 0  ; write FCSE (uTLB invalidate)
MCR    p15, 0, Rtmp2, c7, c5, 4   ; flush prefetch buffer
ADD    r3, pc, #0x24
MCR    p15, 0, r3, c7, c13, 1     ; prefetch I-cache line
```

 © Copyright ARM Limited 2010. All rights reserved.

```
 MCR    p15, 0, rx, c7, c5, 1      ; invalidate I-cache using MVA
[MCRR   p15, 0, ry, rx, c5         ; invalidate I-cache range]
 MSR    cpsr_cx, Rtmp1             ; local_irq_restore
 NOP
 NOP
 NOP
 NOP
 NOP
 NOP
 NOP
```

This workaround sequence has been updated in version 22 of this document to unify it with the corresponding workaround specified in the ARM1176 Errata List for erratum 371367. If you have implemented an earlier version of this workaround then you do not need to update it.

Note that if your system is using the FCSE, you can use the following two instructions to invalidate the microTLB:

```
MRC  p15, 0, Rtmp2, c1, c0, 0 ; read the CP15 Control register
MCR  p15, 0, Rtmp2, c1, c0, 0 ; write the CP15 Control register
```

Workaround 3

Use the Invalidate Entire Instruction Cache instruction instead. You must use the workaround documented in erratum 411920 in order to safely perform this operation.

 © Copyright ARM Limited 2010. All rights reserved.

380532: Load Multiple to the PC can be corrupted or result in deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1, Fixed in r1p2.

Description

In ARM1136J(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.

 © Copyright ARM Limited 2010. All rights reserved.

Implications

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

The code sequence needed to generate this errata 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 cannot happen in Thumb code, as it requires a conditional Load or Store. However, compiled Thumb code can use libraries written using ARM code, and those libraries may contain the failing sequence.

The circumstances for this errata are unusual due to the 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 errata is triggered the result is either data corruption, or a deadlock.

Workaround

It should be noted that the workaround for this is the same as that for errata 364296. As a result, any implementations before r0p5 of ARM1136J(F)-S should be employing a workaround to this problem already, and no additional workaround will be required.

To ensure that this errata 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 mode, 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.

 © Copyright ARM Limited 2010. All rights reserved.

395242: VFP may fail to prevent a load instruction overwriting the source operand of an exceptional data processing instruction

Status

Affects: product ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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

 © Copyright ARM Limited 2010. All rights reserved.

- 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 Technical Reference Manual. 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.

 © Copyright ARM Limited 2010. All rights reserved.

397185: VFP or GCP instruction that stalls may not trace its data with the ETM

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

397187: Interrupts from VIC interface may be taken multiple times

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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 ARM1136 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 synchronously.

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:

 © Copyright ARM Limited 2010. All rights reserved.

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 ARM1136J(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.

 © Copyright ARM Limited 2010. All rights reserved.

399234: Data cache entry can be incorrectly marked as dirty

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2, Fixed in r1p3.

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, it is possible for the dirty bits for the new entry to be incorrectly set and mark the entry as dirty.

If the entry is a Write-Back line then on eviction it will be copied back to external memory. In the majority of cases this is acceptable. If software relies on it not becoming dirty, such as when an external agent updates locations that are also held in the cache then it is possible for the eviction to overwrite the new data.

If the entry is a Write-Through line then on eviction it is possible for old data to overwrite the most recent data written by the core to memory.

Conditions

1. A write-back entry is present in the data cache.
2. A write to that cache entry is present in the level 1 write buffer.
3. A read of a cachable address results in a cache miss and the eviction of the write back cache entry.
4. A write to an unrelated cache entry pushes the head entry from the write buffer after the inefill for the write through address has started.
5. The first piece of data for the new cache line is returned before the eviction of the old cache line is complete.

- Data is returned in less than 10 processor clock cycles if HUM is disabled (less than 8 wait cycles at 1:1 clock ratio)
- Data is returned in less than 15 processor clock cycles if HUM is enabled (less than 13 wait cycles at 1:1 clock ratio)

Implications

This behavior will result in a clean line being marked as dirty. In most situations, this does not matter, but there are two cases where this is important:

1. The line is a Write-Back line, but software is relying that it cannot become dirty to avoid needing to clean it. This is the case where locations are marked as Read-Only and so are known to be clean or the case where an external agent, such as DMA will be updating locations that are also held in the cache.
2. The line is a Write-Through line.In systems that support both Write-Back and Write-Through memory, it is possible for a store to a Write-Through address to update the level 2 memory before the cache entry is updated. If the Write-Through entry was incorrectly marked as dirty, then it will be copied back to

 © Copyright ARM Limited 2010. All rights reserved.

external memory and if it had not been updated before the eviction took place, the old data will be written to external memory. The effect of this erratum is very occasionally to corrupt stored data in write through regions.

Workaround

The erratum may be avoided by ensuring that there are sufficient wait cycles to allow the cache line eviction to complete before the first data item is returned for a linefill (at least 8 wait cycles if HUM is disabled, or at least 13 wait cycles if HUM is enabled).

To avoid the corruption of Write-Through locations all cachable memory should be marked as a single type, either Write-Back or Write-Through.

To avoid problems with Write-Back locations being masked as dirty when being updated by an external agent, the locations should be cleaned and/or invalidated from the cache before the external agent updates memory, even if the location is Read-Only.

 © Copyright ARM Limited 2010. All rights reserved.

399301: Interrupted clean and invalidate operation may not clean data in low interrupt latency configuration

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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, c14`) 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, c14`) 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.

 © Copyright ARM Limited 2010. All rights reserved.

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. The undocumented CP15 Auxiliary Control bit[31] can be set, while bit 21 in the CP15 Control Register remains set. This keeps the precise data abort functionality for low interrupt latency operation, but increases the worst case interrupt latency.
3. Interrupts can be disabled for the duration of the clean and invalidate instruction by setting the I and F bits in the CPSR.
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.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

403345: Jazelle Security

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

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

Conditions

ARM core is executing Java code in Java state.

Implications

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

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

Workaround

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

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

 © Copyright ARM Limited 2010. All rights reserved.

405875: Interrupted Invalidate Instruction Cache by Index might corrupt cache

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3, Fixed in r1p4.

Description

The ARM1136 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

An Invalidate Instruction Cache Line Using Index operation can cause other cache lines within the same cache line group to become valid:

- Cache lines which have previously been invalidated by an Invalidate Entire Instruction Cache operation might become valid, causing stale instructions to be executed by the processor.
- Cache lines which have never been valid might become valid, causing Unpredictable behaviour.

Conditions

The following conditions must occur at the same time:

1. An instruction cache line fill is in progress (this can be to any line).
2. None of the cache lines within the cache line group containing the cache line being invalidated have been valid since reset or the last Invalidate Entire Instruction Cache operation.
3. An Invalidate Instruction Cache Line Using Index operation is executed.
4. An interrupt occurs.

Implications

If the Invalidate Instruction Cache Line Using Index operation is used while interrupts are enabled, the instruction cache can become corrupted.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

There are two possible workarounds.

Workaround 1

Replace the Invalidate Instruction Cache Line Using Index operation with the following sequence of instructions:

```
MRS    Rtmp, cpsr                ; save cpsr
CPSID  ifa                       ; disable interrupts
MCR    p15, 0, Rn, c7, c5, 2     ; invalidate by index
MSR    cpsr_cx, Rtmp             ; restore interrupts
```

Note that you might also need to use the Stop Prefetch Range instruction because of erratum 415622. See the workaround documented in that erratum for more information.

Workaround 2

Use the Invalidate Entire Instruction Cache operation instead. You must use the workaround sequence documented in erratum 411920 to perform this operation safely.

 © Copyright ARM Limited 2010. All rights reserved.

406973: CLREX instruction might be ignored during data cache line fill

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

The CLREX instruction was first implemented in r1p0 of ARM1136J-S and ARM1136JF-S. This instruction clears the local exclusive access lock used by the LDREX and STREX instructions.

The CLREX instruction can sometimes fail to clear the local exclusive access lock, if the data cache is loading a cache line at the same time that the instruction is being executed.

Conditions

An instruction initiates a load to a cache line with the same cache index as the address of the CLREX instruction. The CLREX instruction is then executed while the data cache line fill is still in progress.

Implications

The CLREX instruction might occasionally have no effect.

Workaround

There are two possible workarounds:

1. Use an STREX to a dummy location instead, as recommended for revisions prior to r1p0.
2. Precede the CLREX instruction with a DSB (Data Synchronization Barrier) operation as discussed below.

You can safely use the CLREX instruction if it is preceded by a DSB operation, and interrupts are disabled. This ensures that any data cache line fill is not in progress when the CLREX is executed. Interrupts must be disabled because if an interrupt occurs between the DSB and the CLREX, it could start a new data cache line fill.

The recommended code sequence is as follows:

```
MRS Rtmp, cpsr
CPSID ifa
DSB
CLREX
MSR cpsr_cx, Rtmp
```

 © Copyright ARM Limited 2010. All rights reserved.

408022: Canceled write to the Context ID register might update the ASID

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3, Fixed in r1p4.

Description

The Context ID register is updated using the following instruction:

```
MCR p15, 0, <Rd>, c13, c0, 1
```

Bits [7:0] of this register contain the ASID, which is used by the MMU. The ASID is stored in multiple places in ARM1136. Because of pipelining effects, an update to the ASID can be cancelled when the update is interrupted or is in the shadow of a branch or an exception. Due to this erratum, the update of the copy of the ASID used by the MMU might not be cancelled.

Even in the presence of the erratum, the value of the Context ID register that is returned by reading the register is not corrupted.

Conditions

The erratum can be seen only in privileged code running in ARM state, and in operating systems which use non-global pages. The erratum occurs where an MCR instruction is fetched into the pipeline and advances to Shift pipeline stage before the branch or exception cancels the execution of the MCR instruction. This can occur only in a limited number of practical cases:

1. Where the MCR instruction immediately follows a conditional branch or is the target of a conditional branch OR
2. Where a data value, such as a literal pool value, is interpreted as an MCR instruction and the instruction(s) immediately before MCR instruction is one of
   a. A data-processing instruction (other than an unshifted MOV) that writes to the PC
   b. A data-processing instruction that doesn't write the PC but sets the flags followed by `BX Rn` or `MOV Pc, Rn`
   c. SVC instruction
3. Where an interrupt is taken immediately before the MCR.

Implications

For cases 1 and 2 above, the execution of the MCR results in a corruption of the ASID. If the operating system is using any non-global pages, this can lead to incorrect execution if the corrupted ASID is used to access memory locations held in non-global pages. As the erratum only occurs with privileged code, this does not have any security implications.

The case 2 scenario is most likely to occur in compiled code (or realistic assembly code) with Switch tables when using compilers that implement Switch tables in the following form:

```
ADD  PC, PC, Rn
```

 © Copyright ARM Limited 2010. All rights reserved.

```
DCB  offset-to-case-1
DCB  offset-to-case-2
DCB  offset-to-case-3
DCB  offset-to-case-4
```

While the ADD is being executed, the 4 bytes corresponding to the DCB statement will be interpreted as an instruction. If those 4 bytes are a valid encoding for the Context ID update instruction (32'h_xe0dxf30) then the erratum might occur. It should be noted that there is a 16 million to one chance of 4 random bytes of data corresponding to a valid encoding for the Context ID update.

At present, neither the ARM toolkit nor GCC will implement switch tables in this form, though it is not known what approach is used by other compilers.

Case 3 has no implications for normal code.

Workaround

For case 1 above, the problem can be worked around by the insertion of NOP before the MCR instruction where the MCR is the target of a conditional branch or immediately follows a conditional branch.

For case 2, privileged code must be scanned using a scanning script available from ARM Limited. In the very unlikely event that the relevant code sequence is found, the data values should be changed or placed in different locations to avoid the problem. In the case of the switch table, this would involve the insertion of NOP at the target of the offsets.

Case 3 should not require a workaround.

 © Copyright ARM Limited 2010. All rights reserved.

408660: VFP and GCP instructions are not executed in debug state when the J or T bit is set

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

The ACPINSTRV signal is an output from the ARM1136 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 J or 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 J or 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 J and T bits are not set. The debugger can restore the previous values of the J and T bits if necessary after the coprocessor instructions have been executed.

© Copyright ARM Limited 2010. All rights reserved.

411920: Invalidate Entire Instruction Cache operation might fail to invalidate some lines if coincident with linefill

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3, Fixed in r1p4.

Description

The ARM1136 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.
   - The processor executes a Prefetch Instruction Cache Line or Prefetch Instruction Cache Range operation.
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

 © Copyright ARM Limited 2010. All rights reserved.

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

The prefetch instruction cache prefetch range operation must not be used. If there is a danger of user code using this functionality, the operation can be prevented by setting the Auxiliary Control Register RV bit (bit[5]). This causes an undefined instruction trap if they are used. This will also prevent the use of the instruction cache invalidate range operation.

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 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.

 © Copyright ARM Limited 2010. All rights reserved.

- 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.
  - In ARM1156 and ARM1176 implementations, 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. This is not possible in ARM1136 due to the number of prefetch buffers implemented in ARM1136, but is considered to ensure that a single workaround sequence can be used which applies to ARM1136, ARM1156 and ARM1176 family processors.
  - It is not possible for a fourth linefill to be started in any ARM1136, ARM1156 and ARM1176 family processor, due to the number of prefetch buffers implemented in those processors.
- 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.

 © Copyright ARM Limited 2010. All rights reserved.

424067: An interrupted Invalidate TLB Entry on ASID Match operation can result in microTLB corruption

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

The Invalidate Instruction/Data/Unified TLB Entry on ASID Match operation causes a number of lookups into the TLB to determine whether each TLB entry should be invalidated. If another TLB lookup is made at the same time, it is delayed until the invalidate operation has completed. If an interrupt occurs at the same time, the invalidate operation might be terminated before it completes, allowing the lookup to proceed. If the VIC is enabled, the lookup might return the wrong data.

This can occur if the TLB lookup is caused by an instruction microTLB miss, or a DMA access. This will result in corruption of the instruction microTLB, or incorrect DMA operation.

Conditions

The following conditions must occur at the same time:

1. An Invalidate Instruction/Data/Unified TLB Entry on ASID Match operation is in progress.
2. A TLB lookup is caused by one of the following conditions:
   a. An instruction microTLB miss. This can be caused by the prefetch unit prefetching instructions in a different page.
   b. A DMA access.
3. An interrupt is taken.
4. The VIC is enabled.
5. The MMU is enabled.

Implications

An Invalidate Instruction/Data/Unified TLB Entry on ASID Match operation might cause instruction fetches or DMA accesses to behave as if they were issued to the wrong virtual address. The physical address and page attributes corresponding to a different virtual address will be used.

Workaround

Use the Invalidate Entire TLB operation instead.

Replace, where CRm is c5, c6 or c7:

```
MCR    p15, 0, Rn, c8, CRm, 2      ; invalidate TLB entry on ASID match
```

with:

```
MOV    Rn, #0
```

 © Copyright ARM Limited 2010. All rights reserved.

```
MCR    p15, 0, Rn, c8, CRm, 0      ; invalidate entire TLB
```

 © Copyright ARM Limited 2010. All rights reserved.

424865: Prefetch Instruction Cache Range can cause incorrect operation

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

This erratum obsoletes erratum 415662, which has been removed from this document.

Prefetch Instruction Cache Range is a non-blocking operation that prefetches a range of instruction cache lines while other instructions execute at the same time. This might cause incorrect behaviour when certain other operations or events occur while the prefetch range operation is in progress.

Conditions

A Prefetch Instruction Cache Range operation is executed.

There are a number of different cases where this operation has been found to behave incorrectly which make it impractical to list these conditions.

Implications

Use of the Prefetch Instruction Cache Range operation might result in UNPREDICTABLE behaviour.

Workaround

The Prefetch Instruction Cache Range operation must not be used. If there is a danger of user code using this functionality, the operation can be prevented by setting the Auxiliary Control Register RV bit (bit[5]). This causes an undefined instruction trap if they are used. This will also prevent the use of the instruction cache invalidate range operation.

© Copyright ARM Limited 2010. All rights reserved.

720620: Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 2, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4,r1p5, 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.

 © Copyright ARM Limited 2010. All rights reserved.

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.

 © Copyright ARM Limited 2010. All rights reserved.
Non Confidential

Errata - Category 3

303162: Supersections can return incorrect physical address when subpages enabled

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The physical address returned to the core memory system for a supersection as a result of a page table walk if subpages are enabled takes bits [23:20] of the physical address from bits [23:20] of the L1 descriptor, rather than bits [23:20] of the virtual address (which would be correct behaviour). This is the value that is held in the uTLB.

The physical address returned to the core memory system for a supersection as a result of a page table walk if subpages are enabled (i.e. CP15 register 1 XP bit – bit [23] - is 0) takes bits [23:20] of the physical address from bits [23:20] of the L1 descriptor, rather than bits [23:20] of the virtual address (which would be correct behaviour. This is the value that is held in the uTLB, and so subsequent hits to the uTLB will have the returned physical address similarly corrupted.

The location stored in the main TLB is not corrupted, so accesses to the range of virtual addresses which are covered by the supersection which result in a uTLB miss but a MainTLB hit will not suffer from this corruption, and subsequent uTLB hits will similarly not be corrupted.

When sub-pages are not enabled this erratum does not occur.

Conditions

1. CP15 register 1 XP bit – bit [23] - is 0.
2. Supersections are being used.

Implications

For all existing operating systems this erratum has no effect.

Since supersections are an implementation defined feature of ARM1136, and are not present on any previous cores, no currently existing operating systems make use of supersections. It is generally expected that future operating systems will use the extended page table format (XP=1), and hence not enable subpages. This erratum does not occur when using the extended page table format.

The use of supersections with subpages enabled will result in incorrect operation unless one of the workarounds below is applied.

Workaround

1. Only use supersections with sub-pages disabled (i.e. with CP15 register 1 XP bit – bit [23] - set to 1).
2. Since supersections cover a 16MB range, and so require 16 separate (and currently identical) entries in the L1 page table, each entry can be programmed with a value to bits [23:20] of the L1 descriptor

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

(which is a SBZ field) corresponding to bits [23:20] of the virtual address that will access it. This value is also bits [5:2] of the address of the L1 descriptor in the page table.

 © Copyright ARM Limited 2010. All rights reserved.
Non Confidential

303163: ETM: A low latency interrupt on a java conditional branch may be incorrectly traced.

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

A low latency interrupt on a java conditional branch may lead to the byte code not being traced and the exception incorrectly marked as cancelling.

When an exception or debug entry is traced it is tagged as either a cancelling or non-cancelling exception. If marked as a cancelling exception then this indicates that the last instruction traced did not complete and will be returned to after the exception. The common case for an exception on an instruction is for the instruction to be broadcast in the trace and then for the exception to be marked as a cancelling exception. This enables the trace tools to determine that the last instruction was not executed.

In low interrupt latency mode interrupts can be taken on an instruction without the instruction being broadcast in the trace. In this situation the exception should be marked as non-cancelling.

Due to this erratum, for a low latency interrupt on a java conditional branch the exception can be incorrectly marked as cancelling.

Conditions

This is an issue with the ETM interface in the ARM1136. Under the following conditions the exception can (with some further pipeline conditions) be incorrectly marked as a cancelling exception when the bytecode has not been traced.

1. Processor configured for low interrupt latency mode.
2. An interrupt is taken.
3. The interrupt is taken on a java conditional branch instruction.

Implications

The ETM might incorrectly report which instruction the low latency interrupt occurs on. The ETM would incorrectly report it has having occurred on the previous instruction. This may either be another java bytecode, or an ARM instruction triggering an entry into java state. Unless tracing around java entry this erratum will have no implications. If the first bytecode executed in java state is a java conditional branch and a low latency interrupt is taken on it then the trace may incorrectly show the instruction triggering entry into java as having been interrupted.

If the previous instruction was another java bytecode then currently this errata will have no implications since currently there is no toolkit support for displaying java trace.

Workaround

After an interrupt the exception handler normally returns to the next instruction to be executed. For a cancelling exception this will be the last instruction traced. TraceEnable must be configured to trace the region of code which had the exception. When exception handler completes and returns to the original code, the trace must be

 © Copyright ARM Limited 2010. All rights reserved.

inspected. If the exception was identified as being cancelling and the last instruction traced is not re-executed, but the following instruction is, then it must have been a non-cancelling exception incorrectly identified as a cancelling.

For the conditions described above, the user must look ahead in the trace to determine if the last instruction before the exception or debug entry was correctly marked as cancelling or non-cancelling. This might not be possible if trace is lost due to ETM FIFO overflow.

 © Copyright ARM Limited 2010. All rights reserved.

303164: Aborts on the first part of a V6 unaligned access to strongly ordered memory may not be reported.

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

This erratum applies to a V6 unaligned write to strongly ordered memory which receives an error from the memory / peripheral and an interrupt request is received when in low latency mode.

This is only a possibility if ARM1136 is low interrupt latency mode, V5 backwards compatible page tables are selected, and V6 unaligned accesses are enabled.

If a store is done to strongly ordered (non-cacheable, non-bufferable) memory which is not aligned to a word address when ARM1136 is in V6 unaligned mode then the store gets split into two memory requests when dispatched to the Load/Store unit. If the first memory request receives an error from the external device then the store should be aborted and the second memory request abandoned.

Due to this erratum if an interrupt is received whilst the first memory request is requesting the write on the external bus then the error received from the bus is incorrectly not reported to the core, and the core will instead take the interrupt. The error is not discarded and is instead reported on the next operation which is a load to the program counter.

In general unaligned accesses should not be performed to strongly ordered memory. This is enforced in the V6 architecture when V6 page tables are enabled. When V6 page tables are enabled an unaligned accesses to strongly ordered memory will take an alignment fault.

Conditions

1. ARM1136 in low interrupt latency mode.
2. ARM1136 in V6 unaligned mode.
3. Backwards compatible page tables selected (subpages enabled).
4. A write to strongly ordered memory which is not to a word aligned address which receives an external error on the first transaction of the unaligned access.
5. A fast interrupt request asserted at the time the first part of the write operation is on the AHB-Lite bus.

Implications

There is no backwards compatibility issue with this errata since it requires the use of V6 unaligned accesses.

It is likely that only driver code will have access to strongly ordered memory, which is likely to be a memory mapped access to a control register of some device. Such accesses should be aligned. Locations in strongly ordered memory are not exclusively devices, however the use of unaligned accesses to strongly ordered memory is discouraged, and are prevented when V6 page tables are enabled.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

It is unlikely that a workaround should be required for this erratum. If a workaround is required then any unaligned store to strongly ordered memory should be split into aligned byte / half word stores.

 © Copyright ARM Limited 2010. All rights reserved.

317045: Bits [9:8] of the Data FSR can be written to 1

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

Bits [9:8] of the Data Fault Status Register will take the last value written to these bits, rather than returning 0.

Bits [9:8] of the Data Fault Status Register are defined in the specification as being “Always Read as 0”. The bits are reset to 0, and will always be set to 1 as a result of an abort in hardware which causes the FSR value to change. However, the Data Fault Status Register can be written directly using the instruction

```
MCR p15, 0, <Rd>, c5, c0, 0 .
```

If bits [9:8] of this register is written to 1, then a read back of this register, using the instruction

```
MRC p15, 0, <Rd>, c5, c0, 0
```

will return the value 1 in these bit positions.

Conditions

1. Bits[9:8] of the Data FSR are programmed to values other than b00.
2. The Data FSR is read back before another abort occurs.

Implications

Operating systems which are reading back the Data FSR to determine if an abort since it was last written, and are relying on the value in the FSR bits[9:8] being 0 on such a read will see a corrupted value if a 1 was written to these locations.

It is unlikely that any code will be writing 1 to these bit positions.

Workaround

Where a read of the Data FSR is using this value, and the value has been written by software which could have set 1 to these bit positions, the values of bits [9:8] in the Data FSR should be masked to 0 if having a 1 in either of those positions will lead to erroneous execution.

 © Copyright ARM Limited 2010. All rights reserved.

317046: Debug DSCR Sticky Bits are cleared when the DSCR is written from Scan

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The DSCR bits [7:6] - the Sticky Precise Data Abort bit and the Sticky Imprecise Data Abort bit – are cleared as part of a write to the DSCR from the scan chain.

Bits [7:6] of the DSCR are specified as being reset by a Scan Chain read of the DSCR bits. This is performed by accessing scan chain 1, loading the IR with INTEST, and going through the DR leg to the DBGTAPSM. Due to this erratum, these bits are also cleared on a Scan Chain write of the DSCR, where the IR has EXTEST loaded.

Conditions

1. Scan chain is being used to access the Debug registers
2. The DSCR is being written to by having IR set to EXTEST.
3. The Sticky bits in the DCSR contain non-zero values

Implications

Any debug code, using the scan chain access to the debug registers, which expects the values of the DSCR to remain unchanged during a write to the DCSR will find those bits unexpectedly cleared. Most code accessing the DSCR will have read that data before writing new values to the register, so it is expected that this is unlikely to have any significant implications, as the read causes the clearing of these bits.

Workaround

The values of the sticky bits should be read, ORed with the previous read values, and held in independent storage before any write to the DSCR is performed. Since the scan chain access always reads the data as part of the DR leg of the DBGTAPSM, this read can be performed as part of the access that performs the write.

 © Copyright ARM Limited 2010. All rights reserved.

317047: ACPFLUSH may flush instructions in a Generic Coprocessor after that coprocessor has been deselected

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

A GCP should not contain any instructions when it is deselected by ACPENABLE. Due to this erratum a GCP may contain an instruction which will be flushed by ACPFLUSH after ACPENABLE deselects the coprocessor.

The ACPENABLE[11:0] bus indicates which is the currently selected coprocessor.

When ACPFLUSH is asserted a coprocessor must flush any coprocessor instructions whose tag is less than or equal the tag indicated on ACPFLUSHT.

It was intended that if ACPENABLE for a coprocessor was low, then that coprocessor would not contain a valid instruction. However due to this erratum a coprocessor may be disabled shortly before ACPFLUSH attempts to flush all outstanding coprocessor instructions from that coprocessor.

Thus a GCP coprocessor must ensure that it flushes any valid coprocessor instructions as required by ACPFLUSH even when it is not currently enabled.

Conditions

The conditions for a flush occurring after a coprocessor has been disabled are as follows:

1. A GCP instruction with a hardware breakpoint set on it.
2. The GCP instruction is either a) Directly followed by a GCP instruction for a different coprocessor, or b) followed by a GCP instruction for a different coprocessor with one intermediate ARM instruction.

Implications

If a GCP does not flush a matching coprocessor instruction whilst ACPENABLE indicates it is disabled, this may lead to a pipeline deadlock in the future since the coprocessor instruction will unexpectedly contain an instruction which should have been flushed.

Workaround

If ACPFLUSH is asserted, a coprocessor must flush any instructions whose tag is less than or equal to the flush tag, whether the coprocessor is enabled or not.

© Copyright ARM Limited 2010. All rights reserved.

317048: Performance Counters May Fail to Interrupt on Roll-over of Double Incrementing Events

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The presence on a counter rollover will be missed by the interrupt generation logic if on the cycle that the rollover of the counter occurs, there is a double increment of the counter, causing the counter to go from 0xFFFFFFFF to 0x00000001.

The performance counters implemented in CP15 count a number of different events. A small number of these events can cause the counters to increment by two on a single cycle, where two instances of that event have occurred in a single cycle. If such a Double Incrementing event causes the counter to rollover, and the counters are set to create an interrupt on the event, then interrupt can fail to be triggered.

The events which cause an increment by two on the counters are:

- 0x7 Instruction Executed, when a folded branch fold is executed with another instruction.
- 0x22 ETMEXTOUT[1] and ETMEXTOUT[0] are both asserted simultaneously.

Conditions

1. PMV0 or PMN1 are programmed to count on events 0x7 or 0x22
2. The IntEn bits (PMNC[5:4]) are set to enable an interrupt
3. The PMUIRQ pin is configured to generate a core interrupt

Implications

A performance count routine running for a very large number of cycles may fail to generate an interrupt at rollover, Since the use of an interrupt is usually to cause the increment of a counter in memory every 2^32 events, the result is to miss an increment to this counter in memory, so undercounting the number of events. Due to the relative infrequency of branch folding, it is expected that the effect of this errata will occur very rarely.

In many practical situations, an error in the event count of 2^32 will result in a very implausible result.

Workaround

A number of software workarounds exist:

1. If the system is also configured to generate an interrupt on the cycle counter rollover (using PMNC[6]), then this interrupt can be used to sample the number in the affected counter, and reset it. Since the number of events per cycle should be less than 1 in most practical situations, this would avoid causing a rollover of the affected counter.
2. Programming both counters to be triggered by the same event, but introducing an offset of 1 in their initial counts will result in the rollover being triggered by at least counter.

© Copyright ARM Limited 2010. All rights reserved.

324515: Vector catch on vector 0x18 while VIC enabled

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When the IRQ bit of the VCR (Vector Catch Register) is set to 1 and the VIC (Vectored Interrupt Controller) is enabled, a vector catch debug event will be incorrectly generated on a branch to address 0x18 (or 0xFFFF0018).

If VCR[6] is set to 1 while the VIC is disabled, the ARM1136 debug logic generates a debug event on any change of program flow to addresses 0x18 or 0xFFFF0018 (when in hivecs mode). If, however, the VIC is enabled then the debug logic should only generate debug events on changes of program flow to the most recent address read through the VIC port.

However, the ARM1136 still generates debug events on addresses 0x18 or 0xFFFF0018 when the VIC is enabled.

Conditions

1. Debug enabled; Monitor or Halting debug-mode
2. VCR[6] set to 1
3. VIC enabled, that is, CP15 VE (bit 24) set to 1
4. Branch to address 0x18 (or 0xFFFF0018 in hivecs mode)

Implications

If a debug session is running on a system that uses the VIC port and the processor branches to address 0x18/0xFFFF0018, a debug event will be incorrectly generated and the processor will either enter debug state or take a debug exception.

However, note that, even if the VIC port is enabled, address 0x18/0xFFFF0018 is likely to be left unused as the OS could not possibly reduce its code size by placing two instructions in between the data abort (0x10/0xFFFF0010) and the FIQ (0x1C/0xFFFF001C) vectors. Also, placing OS code in this vector would comprise compatibility with systems that do not use the VIC port and, therefore, it is not a choice the OS designer will make.

Workaround

No workaround is possible for this errata, however it is not expected that this will generate any restriction on the use of this device.

© Copyright ARM Limited 2010. All rights reserved.

324518: Coprocessors only have user mode privileges until the first branch or mode change instruction.

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

External coprocessors and the VFP instructions will only have user mode privileges after reset until the first branch or mode change has occurred.

Some coprocessors can execute a subset of their instructions only if the processor is in a privileged mode. Due to this erratum, if such an instruction is attempted to be executed after reset, but before any type of branch or mode change instruction, the coprocessor will incorrectly believe it cannot execute that instruction, and will take an undefined instruction fault.

It is not expected that this erratum will be seen in practice, since the normal convention is to place a branch instruction of some type at the reset vector thus preventing this errata from occurring.

Conditions

1. Reset
2. No branch or mode change instruction.
3. Execution of a privileged mode only external coprocessor or VFP instruction.

Implications

It is highly unlikely this erratum has any implications since the instruction at the reset vector is usually a branch instruction of some type (either a LDR PC, or B <addr>).This errata can only occur if a coprocessor instruction is executed before any type of branch instruction.

Workaround

If a workaround is required, then a branch instruction or a mode change instruction should be executed before the privileged mode only coprocessor instruction.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

324519: Possibility of ETM not correctly synchronising instruction and data with processor core at ETM power up.

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When the ETM is powered up it must correctly synchronise with the ARM1136 pipeline before any trace can generated. Due to this erratum it may be possible for the ETM not to synchronise if trace occurs very quickly after enabling the ETM.

The ETM module contains logic to enable it to synchronise the ARM1136 instruction trace to the data trace. If the ARM1136 is executing a multi-cycle instruction at the time the ETM is enabled, the outputs to the ETM unit to enable the ETM to correctly synchronise are incorrect, and the ETM may not correctly synchronise instruction and data trace.

However there are many common cases which will cause the ETM to automatically resynchronise instruction and data trace after this point.. An example of a sequence that normally causes automatic resynchronisation is 3 or more non-memory operations, followed by a load or a store. This erratum will only be visible if tracing starts before the ETM has had a change to resynchronise automatically.

Conditions

1. Use of ETM.
2. ETM being powered up in the background to the ARM1136. i.e. when the ARM1136 is not in debug state.
3. ARM1136 is executing a multi-cycle instruction at the time the ETM is powered up.
4. ETM trace starts before the ETM logic has managed to automatically resynchronise instruction and data streams from the ARM1136.

Implications

If invoked this erratum would lead to the instruction trace stream not being correctly aligned with the data trace stream. If a user observes this then in the worst case they would need to reset the device and try reconnecting the debugger with the ETM unit.

Workaround

Possible workarounds if required are:

1. To enable the ETM unit whilst the processor is in debug state.
2. To re-try connecting the debugger with the ETM unit.
3. After powering up the ETM logic, reset the ARM1136.

 © Copyright ARM Limited 2010. All rights reserved.

325158: FSR content may be incorrectly updated if an interrupt occurs immediately before an aborting load or store

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

In low interrupt latency configuration, if an interrupt is taken immediately before a load or store that would otherwise cause a data abort, the FSR will take the value that would have been set by the data abort of the load or store.

The low interrupt latency configuration enables a mechanism to abandon accesses which have been started but which have not passed a commitment point in their execution. This erratum arises because a load or store, which is abandoned due to the presence of the interrupt, has actually committed a write of the FSR due to a data abort, but instead of taking the data abort, the interrupt is recognised instead. As a result, the FSR is corrupted relative to the value it should take at the precise point in the instruction execution that the interrupt is recognised.

Conditions

1. Low interrupt configuration is enabled
2. An interrupt is recognised by the processor immediately before a load or store which would otherwise cause a data abort

Implications

This erratum is not expected to cause any problems for any systems. At the point that the interrupt is recognised, the FSR value must not contain any state which execution will rely on in the future, as had the interrupt not been recognised, the FSR value would have been updated. As a result, the presentation of an incorrectly updated value for the FSR when the interrupt is taken is not expected to lead to any problems.

Workaround

No workaround is believed to be necessary for this erratum.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

328431: VFP can take an Inexact exception on non-CDP VFP instructions

Status

Affects: product ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4,r1p5, 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.

© Copyright ARM Limited 2010. All rights reserved.

329837: CP15 Instruction Cache Data RAM Read Operation reads only even addresses

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The CP15 Instruction cache data RAM Read operation (MCR p14, 3, Rd, c15, c4, 1) is an implementation specific feature of the ARM1136, and not part of the ARM architecture. It is designed to read the contents of the instruction data RAM to act as an aid to debugging code sequences where there is a possibility that the cache contents are stale, due to self-modifying code. Because of this problem, the word specifier (bit[2] of Rd) is not used to rotate the data returned from the instruction cache RAM before writing the data into the instruction debug cache register, so only the even word numbers can be read in this manner. An attempt to read the odd word specified by {Rd[4:3],1} returns the even word specified by {Rd[4:3},0}.

Conditions

CP15 Instruction Cache Data RAM Read operation is being executed from an odd word number

Implications

This behavior is only intended as an aid to debugging for self-modifying code, and is not used by most debug toolkits. This means that the majority of users will not be affected by this problem. Toolkits that do use this functionality will not be critically impaired, because this is not a key debug feature.

In many self-modifying code sequences, more than one word has been altered as part of the self modifying code. In those cases the fact that the cache contains stale entries can be detected from examining the even numbered words in the instruction cache.

Workaround

No workaround is available for this problem. It is categorised as category 3 because the impact is expected to be low from not having this functionality.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

329838: An exception from Java state can, under extremely rare circumstances, corrupt a CP15 operation run shortly afterwards

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

Under very unusual circumstances determined by the state of the pipeline when Java state is entered, if an exception is taken from within Java state then as part of taking the exception from Java state, a state bit within the pipeline is incorrectly set because of this problem. This state bit exists for optimising CP15 data cache maintenance operations. If this state bit is set in this way and there is a CP15 data cache maintenance operation before the first load or store after taking the exception, then the CP15 data cache maintenance operation will not clear the cache pending write buffer. This can lead to pending writes to the cache being ignored, resulting in data corruption.

Conditions

1. Hardware Java acceleration is being used.
2. The instruction after a BXJ instruction is (or appears to be) a CP15 data cache clean or invalidation operation.
3. The exception handlers that can be called from within Java state perform a CP15 data cache clean or invalidate before performing any loads or stores.

Implications

It is expected that all exception handlers will either perform a load to the PC on exception entry, or will perform loads or stores for stacking/unstacking purposes before performing any cache maintenance operations so it is extremely unlikely that this problem will be seen in any practical situation. Linux and other Operating Systems have loads to the PC as part of their exception entry.

Workaround

In the unlikely event that a workaround is required, two possible approaches exist

1. Perform a load to any location before performing a cache maintenance operation in an exception handler.
2. Place a NOP in the instruction immediately following a BXJ instruction.

 © Copyright ARM Limited 2010. All rights reserved.

344292: Corrupting the TLB can deadlock the processor

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2, Fixed in r1p0.

Description

A programming error can cause the TLB to get corrupted, which can result in a memory access matching two or more TLB entries. In this case it is possible that the core will deadlock.

The ARM architecture specifies that if two or more TLB entries match at any time, the behaviour of the TLB is UNPREDICTABLE.

Conditions

1. A programming error results in the instruction uTLB being corrupted.
2. An instruction fetch hits two or more entries in the instruction uTLB.
3. For the first entry, the memory region is non-sharable and either write back or write through.
4. For the second entry, the memory region is sharable.
5. The instruction fetch hits the L1 cache.

Implications

This erratum is caused by a programming error, where two or more TLB entries relate to overlapping ranges of virtual addresses. The ARM architecture gives strong warnings against making such programming error.

This erratum is not expected to occur when running operating systems and applications. However if it does occur, the core will deadlock as a result.

Workaround

There is no known workaround for this erratum.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

397389: Conditional load or store instructions reading uninitialised condition codes may result in deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

The N, Z, C, and V flags in the CPSR have UNPREDICTABLE values after the ARM1136J(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.

 © Copyright ARM Limited 2010. All rights reserved.

401725: ETM may not gain data synchronization on power up

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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

The following conditions must occur at the same time:

1. The power down bit in the ETM control register is changed from 1 to 0.
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.

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.

 © Copyright ARM Limited 2010. All rights reserved.

407121: Unpredictable self-modifying Jazelle code sequence can cause core to hang when BTAC is on

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4,r1p5, Open.

Description

It is possible for unpredictable code executing Java state instructions to cause the processor to hang and become unresponsive to interrupts, when the BTAC is enabled.

The Branch Target Address Cache (BTAC) stores the target address and target T bit of branches. It is keyed using the source address and source T bit. Entries allocated in ARM state are ignored while in Thumb state, and entries allocated in Thumb state are ignored while in ARM state. No J bit information is stored, and any entry can match while in Java state. This is normally not a problem because:

- Java code is incompatible with ARM and Thumb code. It is never legal to execute the same code in Java state and in ARM or Thumb state.
- It is a requirement that the BTAC is invalidated after changing instruction memory, for example in order to replace Java code with ARM or Thumb code.

It is possible to cause an ARM or Thumb state BTAC entry to hit while in Java state by violating these rules. If the target T bit of the BTAC entry is set, then an instruction will be presented to the processor with the T and J bits set. If this instruction is executed, the processor will hang and will not respond to any interrupts.

Conditions

1. The BTAC must be enabled.
2. A branch must be executed in ARM or Thumb state which branches to Thumb state, causing an allocation to the BTAC. The processor must then switch to Java state and an instruction at the address of the previously executed branch must be executed again. The BTAC must not be invalidated at any point during this sequence.

Implications

It is possible for the process to hang and to stop responding to interrupts if:

- Malicious code is executed in user mode with Jazelle enabled.
- A region of instruction memory is replaced by Java code without invalidating the BTAC.

While it is possible for malicious code executed in a privileged mode to have the same effect, this could also be achieved by the same code disabling interrupts, so this has no implications to most systems.

 © Copyright ARM Limited 2010. All rights reserved.

Workaround

You can prevent malicious code executing in any user mode from exercising this erratum by enabling Jazelle only when running a trusted Java Virtual Machine (JVM). You must not enable Jazelle when executing untrusted code.

There is no workaround which prevents this erratum from occurring in privileged modes.

 © Copyright ARM Limited 2010. All rights reserved.

408023: BTAC invalidate ignored while BTAC disabled

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

If a BTAC flush is attempted while the BTAC is disabled, the instruction will have no effect.

Conditions

An instruction which flushes one or more BTAC entries is executed while the BTAC is disabled.

Implications

This will not be a problem for most systems, because operating systems enable the BTAC during the boot process and then leave it permanently enabled. There is no need to invalidate it while it is disabled.

Workaround

There is no need to work around this erratum.

© Copyright ARM Limited 2010. All rights reserved.

413968: A breakpoint on an instruction executed with the J and T bits set might take the undefined instruction trap

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4,r1p5, Open.

Description

The J and T bits of the CPSR indicate whether the current instruction must be executed in Java or Thumb state respectively. It is possible to set both of these bits simultaneously by setting the J and T bits in the SPSR for the current mode and then switching to User or System mode. The next instruction that is fetched will be executed with both the J and T bits of the CPSR set. If a hardware breakpoint is set on this instruction then a breakpoint debug event should occur. If the instruction is at a vector catch address then a vector catch debug event should occur.

Due to this erratum, an undefined instruction exception may incorrectly occur when this instruction is executed instead of the debug event.

Conditions

1. The processor must be in a mode other than User or System.
2. Both the J and T bits of the SPSR must be set.
3. An instruction must be executed to switch to User or System mode.
4. The next instruction that is executed must either:
   a. have a breakpoint set on it.
   b. be at an address that will be caught by the vector catch register.

Implications

Unless debugging this erratum will have no effect.

When debugging code that sets both the J and T bits of the CPSR simultaneously, the undefined instruction trap may be taken when a breakpoint or vector catch debug event is expected.

Workaround

No workaround is required.

 © Copyright ARM Limited 2010. All rights reserved.

427336: FAR/FSR write immediately following precise abort can corrupt FAR/FSR

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

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
```


© Copyright ARM Limited 2010. All rights reserved.

441930: STC immediately following precise external abort updates data cache

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

An STC instruction that immediately follows in program order a load operation that generates a precise external abort may update the data cache even though the instruction is cancelled as a result of the abort.

Conditions

1. Bit 21 of the CP15 Control register is set.
2. A load operation to normal memory misses the data cache and causes an access to the AHB.
3. The load results in an external abort.
4. The load is immediately followed in program order by an STC instruction.
5. The data for the STC is not immediately available from the requested coprocessor.
6. The STC address hits the data cache

In addition to these conditions there are specific timing requirements between the external bus transfer and the availability of data from the coprocessor for the STC.

Implications

The effects of this erratum are unlikely to be observed in normal function. The precise external abort will cause the abort handler to be executed. On return from the exception handler the STC instruction will be executed again and the cache will be updated as expected.

If this erratum is observed, then the data at the STC address may contain an unexpected data value.

Workaround

No workaround is required for this erratum.

 © Copyright ARM Limited 2010. All rights reserved.

Non Confidential

Errata - Implementation

344291: RAM bit and byte write enables are incorrectly driven

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Impl, Present in: r0p2, Fixed in r1p0.

Description

The ARM1136 provides a write enable signal to each RAM, and byte or bit-write enables for RAM blocks that require byte or bit-writes. If your RAM only has byte or bit-write enable inputs, then the write enable signal can normally be left unconnected.

The Memory Built-In Self Test (MBIST) interface provides the ability to test all of the RAMs in the ARM1136 using an external MBIST controller.

When the MBIST interface is used to test RAMs in the ARM1136, byte and bit write enables can remain asserted even when the write enable signals are inactive. This can cause unwanted writes to RAMs if you use RAMs which only have byte or bit write enable inputs.

As a result, RAMs can get corrupted, which can cause MBIST to fail.

The following RAMs are affected by this erratum:

1. IDataRAM
2. IValidRAM
3. ITCDataRAM
4. ITCValidRAM

Conditions

1. The MBIST interface is used to test RAMs in the ARM1136.
2. RAMs do not have a write enable signal, only byte or bit write enables.

Implications

This erratum only affects implementations of the ARM1136 and it only applies to hardware implementers.

This erratum will cause RAMs to get corrupted when performing MBIST, which will result in MBIST failing.

Workaround

If you are affected by the erratum described above, two work-arounds are possible:

1. Make sure that, when implementing the ARM1136, you use a RAM with a write enable and byte or bit write enables, and connect them by using the provided write enable and bit-write enables.
2. If using RAMs with a write enable is not possible, make sure that byte or bit write enables are gated with the write enable signal. This should be done in all RAMs that are affected by this erratum. For example, in the IValidRAM.v file, create the masked bit-write enable bus and connect it to the RAM:

. . . .

 © Copyright ARM Limited 2010. All rights reserved.

```
wire [7:0]  NewBitWr;
assign NewBitWr = {8{IRamValWrite}} & IRamValBitWr[7:0];
...
RAM uValidRAM(
  .CLK  (RAMCLK),
  .EN   (IRamValEn),
  .ADDR (AddrIn),
  .DI   (IRamValIn),
  .BW   (NewBitWr),
  .DO   (IRamValOut)
  );
```


© Copyright ARM Limited 2010. All rights reserved.

351947: Instruction Cache size of 4 or 8 kB can cause deadlock

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Impl, Present in: r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

This erratum is a consequence of the erratum 349888 - "Enabling the Cache Size Restriction can result in a deadlock."

Operating systems that do not support ARM v6 restrictions on page table mappings can set the CZ bit in the CP15 Auxiliary Control Register in order to restrict the cache size to 16 kB. This will enable the processor to run such operating systems since the restrictions do not apply for caches of 16 kB or smaller.

For processor implementations with the Data Cache size greater than 16 kB and the Instruction Cache size less than 16 kB, setting the CZ bit can result in a deadlock.

Conditions

1. The Instruction Cache size is 4 or 8 kB.
2. The Data Cache size is 64 or 32 kB.

Implications

This erratum means that operating systems, which do not implement ARM v6 restrictions on page table mappings, can not be run on processors with Data Cache size of 64 or 32 kB and the Instruction Cache size of 8 or 4 kB.

This erratum can result in a deadlock.

Workaround

Workaround for this erratum is to implement the processor with cache sizes that do not meet conditions described above.

If the cache sizes meet the above conditions, then the operating system running on the processor must not set the CZ bit.

© Copyright ARM Limited 2010. All rights reserved.

409051: Pipelined reset may cause ATPG DRC errors

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Impl, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4, Fixed in r1p5.

Description

The ARM1136J-S 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 SCANMODE 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. Four of the synchronisers lack the masking logic, which can result in a significant loss of fault coverage.

Implications

If SCANMODE is not asserted for scan test, spurious reset pulses will be generated by the 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

SCANMODE must be asserted for scan test.

© Copyright ARM Limited 2010. All rights reserved.

724703: Peripheral Port Asynchronous clock crossing

Status

Affects: product ARM1136J-S, ARM1136JF-S.

Fault status: Impl, Present in: r0p2,r1p0,r1p1,r1p2,r1p3,r1p4,r1p5, Open.

Description

A combinatorial path exists in control signals crossing the asynchronous clock boundary between HCLKPD and CLKIN.

The L2PnRW signal is generated in the CLKIN domain and used in the HCLKPD domain without being qualified with a synchronised request to indicate that it is valid. Depending on the optimisations performed by the synthesis tool and transition on L2PnRW may cause unexpected behaviour in the peripheral port logic.

Conditions

1. HCLKPD is asynchronous to CLKIN (SYNCENPD and HSYNCENPD are tied LOW)
2. A peripheral port read is followed by a peripheral port write

Implications

If synthesis optimisations allow unqualified transitions on L2PnRW to affect the handshake between the CLKIN and HCLKPD clock domains it is possible for spurious handshake acknowledgments to be generated. This can lead to transactions being lost and the processor deadlocking.

If synthesis optimisations allow transitions of CLKIN domain signals to affect the peripheral port AHB control signals, illegal bus activity may be generated.

Workaround

A system workaround is to ensure that the HCLKPD and CLKIN domains are running synchronously. Any crossing to an asynchronous HCLK domain should be done outside of the ARM1136 processor.

 © Copyright ARM Limited 2010. All rights reserved.