IBM PowerPC® 750CXr RISC Microprocessor

Errata List

DD4.X

Version: 1.1

March 7, 2005

IBM ®

This document is intended for hardware system manufacturers and developers of applications, operating systems, tools, firmware, and other software. It is provided to you to describe conditions under which errata may occur in IBM's hardware and to enable you to conduct your own investigation into how your system and software may be impacted and to integrate workarounds as needed.

This document lists errata that IBM has characterized as of January 2003. IBM does not represent or warrant that this errata list is complete. IBM may add errata to this list in the future. Please check with your IBM marketing representative regularly to verify that you have the most current version of this document.

IBM Microelectronics Systems and Technology Group
2070 Route 52, Bldg. 330
Hopewell Junction, NY 12533-6351

The IBM home page can be found at http://www.ibm.com

The IBM Microelectronics home page can be found at http://www.ibm.com/chips

Title_750CXr_Err_DD4.X.fm
March 7, 2005

IBM PowerPC® 750CXr RISC Microprocessor

Errata List

DD4.X

Version: 1.1

March 7, 2005

This document is intended for hardware system manufacturers and developers of applications, operating systems, tools, firmware, and other software. It is provided to you to describe conditions under which errata may occur in IBM's hardware and to enable you to conduct your own investigation into how your system and software may be impacted and to integrate workarounds as needed.

This document lists errata that IBM has characterized as of January 2003. IBM does not represent or warrant that this errata list is complete. IBM may add errata to this list in the future. Please check with your IBM marketing representative regularly to verify that you have the most current version of this document.

IBM Microelectronics Systems and Technology Group
2070 Route 52, Bldg. 330
Hopewell Junction, NY 12533-6351

The IBM home page can be found at http://www.ibm.com

The IBM Microelectronics home page can be found at http://www.ibm.com/chips

Title_750CXr_Err_DD4.X.fm
March 7, 2005

Preface

This document identifies implementation differences between a referenced version of the IBM PowerPC® 750CXr RISC Microprocessor and its corresponding description in the *IBM PowerPC® 750CXr RISC Microprocessor User's Manual.*

The IBM PowerPC 750 CXr RISC Microprocessor, also called the 750CXr, is identified by the following **Processor Version Register (PVR)** value:

- *Version number* 0x0008, *revision number* 0x3410 (for chip level 4.0)

This errata lists any processor differences from the following documents:

- *PowerPC User Instruction Set Architecture* (Book I) version 1.07
- *PowerPC Virtual Environment Architecture* (Book II) version 1.07
- *PowerPC Operating Environment Architecture* (Book III) version 1.07
- *IBM PowerPC 750 CXr RISC Microprocessor User's Manual*

Summary of IBM PowerPC 750 CXr RISC Microprocessor Errata for DD4.X

#	Problem	Overview	Impact	Work-Around	
					4.0
1	L2 cache invalidate may not work with DPM enabled.	If DPM is enabled during a global invalidate of the L2 cache, the global invalidate may not invalidate all of the L2's tags.	Possible system fail after L2 initialization and start-up.	Turn DPM off during an L2tag invalidate.	Y
2	dcbz that hits in L1 cache may not snoop retry.	If a dcbz hits in the L1 cache, a snoop received at the same time to that address may not be serviced or get retried.	Stale data from system memory may be read by the other bus master, and the line may become valid in multiple caches.	Limit use of dcbz to data that is protected through software synchronization.	Y
3	Segment register contention may cause the use of the wrong segment register	mtsr<in> followed by an instruction causing a page data address translation can cause contention for the segment registers.	Possible access to incorrect real address locations or false translation and data access exceptions.	Insert isync, sc, or rfi between any mtsr<in> and instructions that cause a page data address translation.	Y
4	Stfd of uninitialized FPR can hang part.	A stfd will hang the part, if its source FPR has powered up in a certain state.	All systems using stfd's.	Initialize all FPR's at POR.	Y
5	Double snoop push	Snooping may result with two castouts performed to the same address. The first castout may have stale data, but the second castout will have the most recent data.	May result in data corruption in the system or hang condition.	Use one of the following workarounds: 1) Allow both castouts to occur before using castout data. 2) Operate L2 in write-thru mode.	Y
6	SRR0 can become incorrectly set when Performance/Monitor/Decrementer are enabled	A Performance Monitor Interrupt, taken immediately after a delayed Decrementer exception, can cause SRR0 to become incorrect.	The processor hangs.	When performance monitor interrupt signalling is enabled (MMCR0[ENINT]=1), avoid decrementer interrupts by periodically setting the Decrementer register value to 0x7FFFFFFF. Performance monitor interrupts caused by Time Base bit transitions can be used to mediate this periodic setting of the Decrementer and in some cases to roughly simulate decrementer functionality during this time.	Y
7	A 'global invalidate' of a L2 cache with 'dirty' data can result in a write to an erroneous address.	A 'global invalidate' of a L2 cache with 'dirty' data can result in a burst 'write-with-kill' to an erroneous address.	Data can be overwritten and/or system fails may occur due to the erroneous write.	1) Flush the L2 cache prior to disabling and invalidating it. or 2) Disable the L1 ICache, then disable and invalidate the L2.	Y

Note: Errata 1, 2, 3, and 4 previously existed in the 750CXe 2.4 and 3.1 revisions.
Note: 750CXr DD4.x errata #5, 6, & 7 correspond to 750CXe dd3.x errata #6, 7 and 8.

Application Awareness Note:

The thermal sensor in the Thermal Assist Unit (TAU), in the DD levels covered by this errata, has not been characterized to determine the basic uncalibrated accuracy. The relationship between the actual junction temperature and the temperature indicated by THRM1 and THRM2 is not well known.

IBM recommends that the TAU in these devices be calibrated before use. Calibration methods are discussed in the IBM Application Note ***Calibrating the Thermal Assist Unit in the IBM25PPC750L Processors***. Although this note was written for the 750L, the calibration methods discussed in this document also apply to the 750CXr.

Erratum #1: L2 cache invalidate may not work with DPM

Overview

A “global invalidate” of the L2 cache (initiated by the L2I bit of L2CR) may not invalidate all of the L2 cache if dynamic power management (DPM bit of HID0) is also enabled.

Detailed Description

If dynamic power management is enabled (DPM=1 in HID0), a global invalidate of the L2 cache may not properly invalidate the L2 tag memory during the time that the L1’s data cache is waiting for reload data to be received from system memory. During that time, circuity in the L1 data cache is stopped to conserve power, which inadvertently affects the state machine performing the L2 global invalidate operation.

Projected Impact

System fails may occur due to incorrect data in the L2 cache which should have been invalidated.

Workaround

Turn off DPM during the time that a global invalidate of the L2 cache is being performed. Another solution is to ensure that the processor is in a tight uninterrupted software loop monitoring the end of the global invalidate, so that the L1 data cache will never miss and initiate a reload from system memory during the global invalidate operation.

Status

This errata is present in all silicon revisions. Currently a fix is not planned.

Erratum #2: dcbz that hits in L1 cache may not snoop retry

Overview

If the target address of a **dcbz** instruction hits in the 750CXr L1 cache at the same time that a snoop is received to that address, the 750CXr may not react to the snoop, and may not generate a snoop retry to the other bus master. As a result, the other bus master may continue to read the line from system memory (instead of reading 750CXr‘s more recent copy of the data), resulting in stale data or a cache coherency problem.

Detailed Description

If the target address of a **dcbz** instruction hits in the 750CXr L1 cache, the 750CXr will require 4 internal clock cycles to rewrite the cache line to zeros. On the 1st clock, the line is remarked as valid-unmodified, and on the last clock the line is marked as valid-modified. If a snoop request to that address is received during the middle 2 clocks of the **dcbz** operation, 750CXr will not properly react to the snoop operation or generate an address retry (via the ARTRY_ pin) to the other master. The other bus master will continue to read the data from system memory, and both 750CXr and the other bus master will end up with different copies of the data. In addition, if the other bus master has a cache, the line will end up valid in both caches which is not allowed in 750CXr’s 3-state cache environment.

Projected Impact

For data that is shared in real time, incorrect data and data valid in multiple caches may result, causing possible system failures.

Workaround

Use of **dcbz** must be avoided for data that is shared in real time and which is not protected during writing through higher-level software synchronization protocols (such as semaphores). Use of **dcbz** must be avoided for managing semaphores themselves. (An alternative solution could be to prevent **dcbz** from hitting in the L1 cache by performing a **dcbf** to that address beforehand.)

Status

This errata is present in all silicon revisions. Currently a fix is not planned.

Erratum #3: Segment register contention may cause the use of the wrong segment register

Overview

A **mtsr,mtsrin** operation followed closely by an instruction causing a data address translation using page address translation can cause contention for the segment registers. This contention causes the data address translation to proceed using data from the wrong segment register.

Detailed Description

Data page address translations, that attempt to read a segment register during the same cycle a **mtsr,mtsrin** instruction is writing to any of the segment registers, will cause the translation mechanism to receive the written data instead of the contents of the intended segment register. This can occur if there is no context synchronizing instruction between a **mtsr,mtsrin** and a succeeding data address translation that utilize any of the segment registers.

According to PowerPC Architecture, no context synchronizing instructions are required if the context of the surrounding instruction stream is unaffected by the segment register being altered by the **mtsr**.

If a sequence of instructions contains context-altering instructions and contains no instructions that are affected by any of the context alterations, no software synchronization is required within the sequence.[1]

This problem can, within specific timing windows, cause the incorrect segment data to be used for translation under these circumstances.

Instruction address and block address translations are not susceptible to this problem.

Projected Impact

Affected operations can receive incorrect data address translations resulting in access to incorrect real address locations or false translation and data access exceptions.

Workaround

A context-synchronizing instruction (i.e., **isync**) should be placed between any **mtsr,mtsrin** instructions and succeeding instructions that cause a data address translation to take place utilizing any of the segment registers.

Status

This errata is present in all silicon revisions. Currently a fix is not planned.

1.*PowerPC Operating Environment Architecture Book III-AIM* Version 1.07, Chapter 7, page 75.

Erratum #4: Stfd of uninitialized FPR can hang part

Overview

A **stfd** can cause the part to hang if its source FPR has powered up in a certain state.

Detailed Description

The 64-bit FPRs each have additional internal bits associated with them that specify the type of floating point number that the register contains. These bits get properly set whenever the FPR is loaded. It is possible, however, for the part to power up with the internal bits randomly set, such that the FPR is interpreted as containing a denormalized number, but with the mantissa containing all zeroes. If this random state is stored out with an **stfd** before the internal bits are corrected via an FP load operation, the part will hang searching for a leading '1' in the mantissa.

The **stfd** is the only instruction that causes this behavior.

Note that this problem was discovered when *compiled* code stored out FPRs in preparation for using them as scratch registers early in the boot sequence.

Projected Impact

This affects all systems that use floating point operations.

Workaround

Upon coming out of a Power-On-Reset (POR), initialize all of the FPRs that will be used. The value used for initialization is not important.

Status

This is present in all silicon revisions. Currently a fix is not planned.

Erratum #5: Double Snoop Push

Overview

Snooping may result with two castouts performed to the same address. The first castout may have stale data, but the second castout may have the most recent data.

Detailed Description

For the double snoop push to occur, the following sequence of events must occur:

1. Cache line ‘A’ is dirty (modified with respect to system memory) in both the L1 data cache and in the L2 cache. This can occur as a result of normal tag re-allocations in the L1 data cache.
2. In the L2 cache, the cache tag for cache in line ‘A’ is then reallocated for another address as a result of normal reloading of instruction fetches or load/store operations. As a result, cache line ‘A’ and at least one other dirty sector for that cache tag is sent from the L2 cache to the bus unit to be written out to memory. During this time, cache line ‘A’ is still valid and modified in the L1 cache.
3. Another bus master then performs a bus transaction which is snooped and hits for for cache line’A’. Internally, the snoop hits in both the L1 cache and in the bus unit’s L2 castout buffer. The correct response in this case would be to cancel the L2 castout in the bus unit and push the line from the L1 cache (which contains the most modified data). However, the presence of more than one sector in the L2 castout buffer results in incorrect behavior. The L2 castout buffer is pushed to the bus, and then the same cache line is again pushed (but with newer data) from the L1 cache.

   The result is that after the snoop appears to have completed on the bus, the snoop push buffer in the bus unit still contains the most modified cache line from the L1 cache, giving the appearance that the processor missed the snoop ‘window of opportunity’. This also blocks the snoop buffer from being available for an immediately subsequent snoop operation.

Projected Impact

If the second snoop push is not allowed to be performed before the original bus master that triggered the snoop is allowed to access system memory, the original bus master will then read stale data from memory instead of the latest data which is still in the processor snoop push buffer. This may result in data corruption in the system.

Also, if a new snoop triggers a snoop push from the processor while the snoop buffer still contains the previous uncompleted second snoop push, the processor will respond to the new snoop by attempting to push the previous snoop address, in order to clear its buffer. If the bus master generating the new snoop does not allow this different address to be pushed, but rather keeps retrying it until the expected new snoop address is pushed instead, a system ‘livelock’ may occur where the new snoop is never completed (usually requiring the system to be reset).

Workaround

Use one of the following workarounds:

1. Allow both snoop pushes to be performed before continuing the alternate master that triggered the snoop. In an MPC106-based system (Grackle) bridge chip, this can be accomplished by setting the CF_LOOP_SNOOP bit in the MPC106's configuration registers.
2. Prevent dirty data from residing in the processor's L2 cache. This can be achieved by operating the L2 cache in write-thru mode by setting the L2WT bit in its L2 configuration register. This workaround works in any system configuration. In systems employing a 750CXr, the MPC106 and a DMA device on the 60x bus which can generate snoops, to the 750CXr, a more narrow workaround could be to operate only those pages addresable by the DMA device in write-thru (or cache inhibit), while still setting the CF_LOOP_SNOOP bit in the MPC106.

Note: In systems where all snoopable memory (such as system memory) is accessed directly and only over 60x bus, the occurrence of a double snoop push would not cause a failure, and a workaround would normally not be required. In this case, the two snoop pushes would get an opportunity to be performed before the alternate master continues on the 60x bus due to normal 60x bus retry protocols.

Status

A fix may be included in a future revision.

Erratum #6: SRR0 can become incorrectly set when Performance Monitor/Decrementer is enabled

Overview

A performance monitor interrupt, taken immediately after a delayed decrementer exception, can cause SRR0 to become incorrect.

Detailed Description

A decrementer exception is signalled while MSR[EE]=0. The performance monitor is also enabled and counting events during this time. Sometime later, MSR[EE] gets set to ‘1’ and the delayed decrementer exception starts to be taken. At the same time, a performance monitor interrupt is taken, which is a higher priority interrupt. One core cycle after the DEC exception is started, the PM interrupt is taken instead. SRR0 gets set to 0x00000900 - but the DEC exception handler never started, so no state was saved. At this point, the PM exception has become unrecoverable because it would return to the 0x900 handler and not be able to return to the normal program flow after that point.

Projected Impact

The processor hangs.

Workaround

When performance monitor interrupt signalling is enabled (MMCR0[ENINT]=1), avoid decrementer interrupts by periodically setting the decrementer register value to 0x7FFFFFFF. Performance monitor interrupts caused by Time Base bit transitions can be used to mediate this periodic setting of the decrementer and in some cases to roughly simulate decrementer functionality during this time.

Status

A fix may be included in a future revision.

Erratum #7: A global invalidate of L2 cache with dirty data can result in a write to an incorrect address

Overview

A global invalidate of L2 cache with dirty data can result in a burst write-with-kill to an incorrect address.

Detailed Description

It is possible for the 750CXr processor to issue an incorrect burst write-with-kill transaction to address 0xFFFFFXXX. This incorrect transaction can occur only if:

1. The L1 Icache is enabled
2. Code is executed that disabled and invalidates the L2 cache
3. The L2 cache had dirty or modified data not yet written to the 60x bus

The incorrect transaction can occur when a code stream that disables and invalidates the L2 cache executes. An instruction pre-fetch just before the L2 cache is disabled allocates in the L2 cache, replacing a modified line and causing a castout to be queued. If the L2 cache invalidate starts before the castout address phase is started on the 60x bus, the castout address will not work.

Projected Impact

Data integrity at address 0xFFFFFXXX and/or system fails may occur due to the incorrect write. In some applications the memory controller will respond to the transaction with a TEA causing a processor checkstop or machine check.

Workaround

1. It is always recommended to flush the L2 cache prior to disabling and invalidating it to prevent loss of modified data. The flush operation is described in the 750CXr user’s manual.
2. To avoid the overhead of the flush operation when the modified data in the L2 cache can be ignored, another option is to disable the L1 cache and then disable and invalidate the L2 as follows:

```
isync
hid0(ice)<-0 /* disable Icache */
isync
….
l2cr(l2e)<-0    /* disable L2 Cache */
....
l2cr(l2i)<-1 /* invalidate L2 Cache */
...
isync
hid0(ice)<-1 /* re-enable Icache if needed */
isync
...,
wait for l2cr(ip)==0 /* wait for L2 Cache invalidate */
...
```

This will ensure that a pending L2 castout has been serviced by the 60x bus. If the L1 DCache is enabled, the user must also disable the L1 DCache prior to the L2 disable and invalidation.

Status

A fix may be included in a future revision.

Preface

This document identifies implementation differences between a referenced version of the IBM PowerPC® 750CXr RISC Microprocessor and its corresponding description in the *IBM PowerPC® 750CXr RISC Micro-processor User's Manual.*

The IBM PowerPC 750 CXr RISC Microprocessor, also called the 750CXr, is identified by the following **Processor Version Register (PVR)** value:

- *Version number* 0x0008, *revision number* 0x3410 (for chip level 4.0)

This errata lists any processor differences from the following documents:

- *PowerPC User Instruction Set Architecture* (Book I) version 1.07
- *PowerPC Virtual Environment Architecture* (Book II) version 1.07
- *PowerPC Operating Environment Architecture* (Book III) version 1.07
- *IBM PowerPC 750 CXr RISC Microprocessor User's Manual*

Summary of IBM PowerPC 750 CXr RISC Microprocessor Errata for DD4.X

#	Problem	Overview	Impact	Work-Around	
					4.0
1	L2 cache invalidate may not work with DPM enabled.	If DPM is enabled during a global invalidate of the L2 cache, the global invalidate may not invalidate all of the L2's tags.	Possible system fail after L2 initialization and start-up.	Turn DPM off during an L2tag invalidate.	Y
2	dcbz that hits in L1 cache may not snoop retry.	If a dcbz hits in the L1 cache, a snoop received at the same time to that address may not be serviced or get retried.	Stale data from system memory may be read by the other bus master, and the line may become valid in multiple caches.	Limit use of dcbz to data that is protected through software synchronization.	Y
3	Segment register contention may cause the use of the wrong segment register	mtsr<in> followed by an instruction causing a page data address translation can cause contention for the segment registers.	Possible access to incorrect real address locations or false translation and data access exceptions.	Insert isync, sc, or rfi between any mtsr<in> and instructions that cause a page data address translation.	Y
4	Stfd of uninitialized FPR can hang part.	A stfd will hang the part, if its source FPR has powered up in a certain state.	All systems using stfd's.	Initialize all FPR's at POR.	Y
5	Double snoop push	Snooping may result with two castouts performed to the same address. The first castout may have stale data, but the second castout will have the most recent data.	May result in data corruption in the system or hang condition.	Use one of the following workarounds: 1) Allow both castouts to occur before using castout data. 2) Operate L2 in write-thru mode.	Y
6	SRR0 can become incorrectly set when Performance/Monitor/Decrementer are enabled	A Performance Monitor Interrupt, taken immediately after a delayed Decrementer exception, can cause SRR0 to become incorrect.	The processor hangs.	When performance monitor interrupt signalling is enabled (MMCR0[ENINT]=1), avoid decrementer interrupts by periodically setting the Decrementer register value to 0x7FFFFFFF. Performance monitor interrupts caused by Time Base bit transitions can be used to mediate this periodic setting of the Decrementer and in some cases to roughly simulate decrementer functionality during this time.	Y
7	A 'global invalidate' of a L2 cache with 'dirty' data can result in a write to an erroneous address.	A 'global invalidate' of a L2 cache with 'dirty' data can result in a burst 'write-with-kill' to an erroneous address.	Data can be overwritten and/or system fails may occur due to the erroneous write.	1) Flush the L2 cache prior to disabling and invalidating it. or 2) Disable the L1 ICache, then disable and invalidate the L2.	Y

Note: Errata 1, 2, 3, and 4 previously existed in the 750CXe 2.4 and 3.1 revisions.
Note: 750CXr DD4.x errata #5, 6, & 7 correspond to 750CXe dd3.x errata #6, 7 and 8.

Application Awareness Note:

The thermal sensor in the Thermal Assist Unit (TAU), in the DD levels covered by this errata, has not been characterized to determine the basic uncalibrated accuracy. The relationship between the actual junction temperature and the temperature indicated by THRM1 and THRM2 is not well known.

IBM recommends that the TAU in these devices be calibrated before use. Calibration methods are discussed in the IBM Application Note ***Calibrating the Thermal Assist Unit in the IBM25PPC750L Processors***. Although this note was written for the 750L, the calibration methods discussed in this document also apply to the 750CXr.

Erratum #1: L2 cache invalidate may not work with DPM

Overview

A “global invalidate” of the L2 cache (initiated by the L2I bit of L2CR) may not invalidate all of the L2 cache if dynamic power management (DPM bit of HID0) is also enabled.

Detailed Description

If dynamic power management is enabled (DPM=1 in HID0), a global invalidate of the L2 cache may not properly invalidate the L2 tag memory during the time that the L1’s data cache is waiting for reload data to be received from system memory. During that time, circuity in the L1 data cache is stopped to conserve power, which inadvertently affects the state machine performing the L2 global invalidate operation.

Projected Impact

System fails may occur due to incorrect data in the L2 cache which should have been invalidated.

Workaround

Turn off DPM during the time that a global invalidate of the L2 cache is being performed. Another solution is to ensure that the processor is in a tight uninterrupted software loop monitoring the end of the global invalidate, so that the L1 data cache will never miss and initiate a reload from system memory during the global invalidate operation.

Status

This errata is present in all silicon revisions. Currently a fix is not planned.

Erratum #2: dcbz that hits in L1 cache may not snoop retry

Overview

If the target address of a **dcbz** instruction hits in the 750CXr L1 cache at the same time that a snoop is received to that address, the 750CXr may not react to the snoop, and may not generate a snoop retry to the other bus master. As a result, the other bus master may continue to read the line from system memory (instead of reading 750CXr's more recent copy of the data), resulting in stale data or a cache coherency problem.

Detailed Description

If the target address of a **dcbz** instruction hits in the 750CXr L1 cache, the 750CXr will require 4 internal clock cycles to rewrite the cache line to zeros. On the 1st clock, the line is remarked as valid-unmodified, and on the last clock the line is marked as valid-modified. If a snoop request to that address is received during the middle 2 clocks of the **dcbz** operation, 750CXr will not properly react to the snoop operation or generate an address retry (via the ARTRY_ pin) to the other master. The other bus master will continue to read the data from system memory, and both 750CXr and the other bus master will end up with different copies of the data. In addition, if the other bus master has a cache, the line will end up valid in both caches which is not allowed in 750CXr's 3-state cache environment.

Projected Impact

For data that is shared in real time, incorrect data and data valid in multiple caches may result, causing possible system failures.

Workaround

Use of **dcbz** must be avoided for data that is shared in real time and which is not protected during writing through higher-level software synchronization protocols (such as semaphores). Use of **dcbz** must be avoided for managing semaphores themselves. (An alternative solution could be to prevent **dcbz** from hitting in the L1 cache by performing a **dcbf** to that address beforehand.)

Status

This errata is present in all silicon revisions. Currently a fix is not planned.

Erratum #3: Segment register contention may cause the use of the wrong segment register

Overview

A **mtsr,mtsrin** operation followed closely by an instruction causing a data address translation using page address translation can cause contention for the segment registers. This contention causes the data address translation to proceed using data from the wrong segment register.

Detailed Description

Data page address translations, that attempt to read a segment register during the same cycle a **mtsr,mtsrin** instruction is writing to any of the segment registers, will cause the translation mechanism to receive the written data instead of the contents of the intended segment register. This can occur if there is no context synchronizing instruction between a **mtsr,mtsrin** and a succeeding data address translation that utilize any of the segment registers.

According to PowerPC Architecture, no context synchronizing instructions are required if the context of the surrounding instruction stream is unaffected by the segment register being altered by the **mtsr**.

If a sequence of instructions contains context-altering instructions and contains no instructions that are affected by any of the context alterations, no software synchronization is required within the sequence.[1]

This problem can, within specific timing windows, cause the incorrect segment data to be used for translation under these circumstances.

Instruction address and block address translations are not susceptible to this problem.

Projected Impact

Affected operations can receive incorrect data address translations resulting in access to incorrect real address locations or false translation and data access exceptions.

Workaround

A context-synchronizing instruction (i.e., **isync**) should be placed between any **mtsr,mtsrin** instructions and succeeding instructions that cause a data address translation to take place utilizing any of the segment registers.

Status

This errata is present in all silicon revisions. Currently a fix is not planned.

1.*PowerPC Operating Environment Architecture Book III-AIM* Version 1.07, Chapter 7, page 75.

Erratum #4: Stfd of uninitialized FPR can hang part

Overview

A **stfd** can cause the part to hang if its source FPR has powered up in a certain state.

Detailed Description

The 64-bit FPRs each have additional internal bits associated with them that specify the type of floating point number that the register contains. These bits get properly set whenever the FPR is loaded. It is possible, however, for the part to power up with the internal bits randomly set, such that the FPR is interpreted as containing a denormalized number, but with the mantissa containing all zeroes. If this random state is stored out with an **stfd** before the internal bits are corrected via an FP load operation, the part will hang searching for a leading ‘1’ in the mantissa.

The **stfd** is the only instruction that causes this behavior.

Note that this problem was discovered when *compiled* code stored out FPRs in preparation for using them as scratch registers early in the boot sequence.

Projected Impact

This affects all systems that use floating point operations.

Workaround

Upon coming out of a Power-On-Reset (POR), initialize all of the FPRs that will be used. The value used for initialization is not important.

Status

This is present in all silicon revisions. Currently a fix is not planned.

Erratum #5: Double Snoop Push

Overview

Snooping may result with two castouts performed to the same address. The first castout may have stale data, but the second castout may have the most recent data.

Detailed Description

For the double snoop push to occur, the following sequence of events must occur:

1. Cache line ‘A’ is dirty (modified with respect to system memory) in both the L1 data cache and in the L2 cache. This can occur as a result of normal tag re-allocations in the L1 data cache.
2. In the L2 cache, the cache tag for cache in line ‘A’ is then reallocated for another address as a result of normal reloading of instruction fetches or load/store operations. As a result, cache line ‘A’ and at least one other dirty sector for that cache tag is sent from the L2 cache to the bus unit to be written out to memory. During this time, cache line ‘A’ is still valid and modified in the L1 cache.
3. Another bus master then performs a bus transaction which is snooped and hits for for cache line’A’. Internally, the snoop hits in both the L1 cache and in the bus unit’s L2 castout buffer. The correct response in this case would be to cancel the L2 castout in the bus unit and push the line from the L1 cache (which contains the most modified data). However, the presence of more than one sector in the L2 castout buffer results in incorrect behavior. The L2 castout buffer is pushed to the bus, and then the same cache line is again pushed (but with newer data) from the L1 cache.

   The result is that after the snoop appears to have completed on the bus, the snoop push buffer in the bus unit still contains the most modified cache line from the L1 cache, giving the appearance that the processor missed the snoop ‘window of opportunity’. This also blocks the snoop buffer from being available for an immediately subsequent snoop operation.

Projected Impact

If the second snoop push is not allowed to be performed before the original bus master that triggered the snoop is allowed to access system memory, the original bus master will then read stale data from memory instead of the latest data which is still in the processor snoop push buffer. This may result in data corruption in the system.

Also, if a new snoop triggers a snoop push from the processor while the snoop buffer still contains the previous uncompleted second snoop push, the processor will respond to the new snoop by attempting to push the previous snoop address, in order to clear its buffer. If the bus master generating the new snoop does not allow this different address to be pushed, but rather keeps retrying it until the expected new snoop address is pushed instead, a system ‘livelock’ may occur where the new snoop is never completed (usually requiring the system to be reset).

Workaround

Use one of the following workarounds:

1. Allow both snoop pushes to be performed before continuing the alternate master that triggered the snoop. In an MPC106-based system (Grackle) bridge chip, this can be accomplished by setting the CF_LOOP_SNOOP bit in the MPC106's configuration registers.
2. Prevent dirty data from residing in the processor's L2 cache. This can be achieved by operating the L2 cache in write-thru mode by setting the L2WT bit in its L2 configuration register. This workaround works in any system configuration. In systems employing a 750CXr, the MPC106 and a DMA device on the 60x bus which can generate snoops, to the 750CXr, a more narrow workaround could be to operate only those pages addresable by the DMA device in write-thru (or cache inhibit), while still setting the CF_LOOP_SNOOP bit in the MPC106.

Note: In systems where all snoopable memory (such as system memory) is accessed directly and only over 60x bus, the occurrence of a double snoop push would not cause a failure, and a workaround would normally not be required. In this case, the two snoop pushes would get an opportunity to be performed before the alternate master continues on the 60x bus due to normal 60x bus retry protocols.

Status

A fix may be included in a future revision.

Erratum #6: SRR0 can become incorrectly set when Performance Monitor/Decrementer is enabled

Overview

A performance monitor interrupt, taken immediately after a delayed decrementer exception, can cause SRR0 to become incorrect.

Detailed Description

A decrementer exception is signalled while MSR[EE]=0. The performance monitor is also enabled and counting events during this time. Sometime later, MSR[EE] gets set to ‘1’ and the delayed decrementer exception starts to be taken. At the same time, a performance monitor interrupt is taken, which is a higher priority interrupt. One core cycle after the DEC exception is started, the PM interrupt is taken instead. SRR0 gets set to 0x00000900 - but the DEC exception handler never started, so no state was saved. At this point, the PM exception has become unrecoverable because it would return to the 0x900 handler and not be able to return to the normal program flow after that point.

Projected Impact

The processor hangs.

Workaround

When performance monitor interrupt signalling is enabled (MMCR0[ENINT]=1), avoid decrementer interrupts by periodically setting the decrementer register value to 0x7FFFFFFF. Performance monitor interrupts caused by Time Base bit transitions can be used to mediate this periodic setting of the decrementer and in some cases to roughly simulate decrementer functionality during this time.

Status

A fix may be included in a future revision.

Erratum #7: A global invalidate of L2 cache with dirty data can result in a write to an incorrect address

Overview

A global invalidate of L2 cache with dirty data can result in a burst write-with-kill to an incorrect address.

Detailed Description

It is possible for the 750CXr processor to issue an incorrect burst write-with-kill transaction to address 0xFFFFFXXX. This incorrect transaction can occur only if:

1. The L1 Icache is enabled
2. Code is executed that disabled and invalidates the L2 cache
3. The L2 cache had dirty or modified data not yet written to the 60x bus

The incorrect transaction can occur when a code stream that disables and invalidates the L2 cache executes. An instruction pre-fetch just before the L2 cache is disabled allocates in the L2 cache, replacing a modified line and causing a castout to be queued. If the L2 cache invalidate starts before the castout address phase is started on the 60x bus, the castout address will not work.

Projected Impact

Data integrity at address 0xFFFFFXXX and/or system fails may occur due to the incorrect write. In some applications the memory controller will respond to the transaction with a TEA causing a processor checkstop or machine check.

Workaround

1. It is always recommended to flush the L2 cache prior to disabling and invalidating it to prevent loss of modified data. The flush operation is described in the 750CXr user’s manual.
2. To avoid the overhead of the flush operation when the modified data in the L2 cache can be ignored, another option is to disable the L1 cache and then disable and invalidate the L2 as follows:

```
isync
hid0(ice)<-0 /* disable Icache */
isync
….
l2cr(l2e)<-0    /* disable L2 Cache */
....
l2cr(l2i)<-1 /* invalidate L2 Cache */
...
isync
hid0(ice)<-1 /* re-enable Icache if needed */
isync
...,
wait for l2cr(ip)==0 /* wait for L2 Cache invalidate */
...
```

This will ensure that a pending L2 castout has been serviced by the 60x bus. If the L1 DCache is enabled, the user must also disable the L1 DCache prior to the L2 disable and invalidation.

Status

A fix may be included in a future revision.

Revision Log

Revision Date	Contents of Modification
June 4, 2003	Version1.0. Preliminary draft.
July 24, 2003	Version1.0. Incorporated reviewer comments.
August 1, 2003	Version1.0. Incorporated reviewer comments.
September 24, 2003	Version1.0. Updated based on legal review
March 7 2005	Version 1.1 New Release Removed “Preliminary” from Document. Updated legal information page.

Revision Log

Revision Date	Contents of Modification
June 4, 2003	Version1.0. Preliminary draft.
July 24, 2003	Version1.0. Incorporated reviewer comments.
August 1, 2003	Version1.0. Incorporated reviewer comments.
September 24, 2003	Version1.0. Updated based on legal review
March 7 2005	Version 1.1 New Release Removed “Preliminary” from Document. Updated legal information page.