IBM Gekko RISC Microprocessor

Errata List

Release DD2.0/2.4 Chips

Version 1.1 for DD1.0, 2.0, 2.3, 2.4

October 16, 2000

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY
12533-6531

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

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

Preface

This document identifies implementation differences between a referenced version of the Gekko processor and the description of the Gekko processor contained in the *IBM Gekko RISC Microprocessors User's Manual*.

The Gekko processor is identified by **Processor Version Register (PVR)** values:

- *Version Number* 0x7000, *Revision Number* 0x0100 (for chip level 1.0, with fuse blown)
- *Version Number* 0x0008, *Revision Number* 0x0100 (for chip level 1.0, with no fuse blown)
- *Version Number* 0x7000, *Revision Number* 0x0100 (for chip level 2.0, with fuse blown)
- *Version Number* 0x0008, *Revision Number* 0x0100 (for chip level 2.0, default, no fuse blown)
- *Version Number* 0x0008, *Revision Number* 0x3203 (for chip level 2.3e, initial lot of 50)
- *Version Number* 0x0008, *Revision Number* 0x3213 (for chip level 2.3e)
- *Version Number* 0x0008, *Revision Number* 0x3204 (for chip level 2.4)

This document 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 Gekko RISC Microprocessors User's Manual

Table 1: Summary of IBM Gekko RISC Microprocessor Version 1.1 for DD1.0, 2.0, 2.3, 2.4 Errata

Note: The Gekko errata listed below correspond to the 750-PID8t errata in following manner:
1 = PID8t Erratum #4, 2 = PID8t Erratum #7, 3 = PID8t Erratum #9, 4 = PID8t Erratum #12, 5 = PID8t Erratum #16.

#	Problem	Description	Impact	Solution(s)	Applicable to Version			
					DD 1.0	DD 2.0	DD2.3	DD2.4
1	L2 cache invalidate may fail 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	Y	Y	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	Y	Y	Y
3	Segment register updates may corrupt data translation.	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	Y	Y	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	Y	Y	Y
5	Snoop push may occur twice.	When Gekko snoops a bus transaction and does a snoop push to system memory as a result, it may push the cache line twice. The second push will contain the most recent (most coherent) data.	Data corruption if the 2nd snoop push is not allowed to occur before the alternate master resumes, or possible system livelock if system does not allow the 2nd snoop to occur before a new snoop push is expected.	Allow both snoops to occur before resuming the alternate master, or operate L2 cache in write-through mode.	Y	Y	Y	Y
6	DMA QCNT decrements before load data is in the cache.	In some cases, the DMA QCNT field of the HID_G register decrements before the DMA load data is in the cache.	Possible DMA and system failure due to wrong transfer count.	Code all DMA requests to hit the data cache.	Y	N	N	N
7	Write Gather is not pipelined.	Consecutive bus transactions for the write gather pipe are not pipelined.	Possible decreased bus performance.	None.	Y	N	N	N
8	Unaligned stfd may cause processor hang.	A store float double (quantized store) can collide with a tablewalk operation causing a processor hang condition.	No impact if segmented address translation is not used. If translation is used a possible hang condition may result.	Avoid using segmented address translation. If segmented address translation is used, then floating point double operands must be double word-aligned.	Y	N	N	N
9	Invalid transaction type with DCBZ(M=0) during DMA.	DCBZ with the M bit of the WIMG bits equal to zero may produce an invalid transaction type if the DCBZ causes a copyback while a DMA transaction is in progress	Bus contention problems may occur as a result of this error.	Setup all DCBZ instructions to use M=1 setting for WIMG bits	Y	N	N	N

Table 1: Summary of IBM Gekko RISC Microprocessor Version 1.1 for DD1.0, 2.0, 2.3, 2.4 Errata

Note: The Gekko errata listed below correspond to the 750-PID8t errata in following manner:
1 = PID8t Erratum #4, 2 = PID8t Erratum #7, 3 = PID8t Erratum #9, 4 = PID8t Erratum #12, 5 = PID8t Erratum #16.

#	Problem	Description	Impact	Solution(s)	Applicable to Version			
					DD 1.0	DD 2.0	DD2.3	DD2.4
10	DMA hang due to L2 request queue full	A DMA load request gets dropped resulting in a processor hang	If this situation occurs, the processor will hang waiting for the second DMA load to complete.	Ensure the L2 request queue is cleared with a sync instruction before initiating a DMA load, or run with the L2 cache disabled	Y	N	N	N
11	Store lost due to queued L/S requests during DMA	A processor store gets lost while running DMA because a copy back from the data cache does not occur	Store data will be lost	None	Y	N	N	N
12	L2 reload data incorrect during DMA	Data for an L2 reload due to an instruction fetch request is written incorrectly while DMA is active	Incorrect instruction sequence may result if the line that was loaded in the l2 is accessed again	Run with the L2 cache disabled.	Y	N	N	N
13	RESERVE register not updated during DMA	The Reserve register used by the lwarx instruction is not updated correctly in some cases.	The lwarx instruction does not function correctly.	When DMA is active avoid using the LWARX instruction.	Y	N	N	N
14	D TLB can be corrupted during DMA	TLB miss processing by the hard-ware tablewalker can result in TLB corruption if run concurrently with DMA.	Paging with DMA will not work	Avoid using the TLB with DMA operations	Y	N	N	N
15	DCBT/DCBTST may cause a hang.	A hang condition may occur if the cache is processing a miss and a DCBT/DCBTST is received.	Possible system hang if the DCBT/ DCBTST instructions are used prevents use of these instructions	None	Y	N	N	N
16	DMA store bus utilization is not optimal	DMA back to back write perfor-mance is not optimal.	Bus utilization and performance is not optimum	None	Y	N	N	N
17	Incorrect Load completion during single step	The load/store unit is incorrectly completing a load in a single step mode	Software debug activities using a single step mode will potentially hang or give incorrect results.	None	Y	N	N	N
18	Write pipe request gets repeated	A store request gets queued mul-tiple times in the write gather pipe queue	Data integrity problems may occur when using the write gather pipe with the data cache enabled.	Disable the data cache when using the write gather pipe.	Y	N	N	N
19	Store dependency results in stale data	The load/store unit is storing stale data for floating point and quan-tized stores when there is a store dependency condition	Unaligned floating point and quantized stores may result in incorrect data written to memory.	Keep all floating point double preci-sion and quantized stores double word aligned or keep unaligned FP and PSQ store accesses in the same page, not across page boundaries.	Y	N	N	N
20	DMA load data integrity problem.	In some cases, an extra DMA load request is issued to the BIU. This causes data cache to be out of sync with the BIU.	Data integrity problems may occur when using DMA loads.	None	Y	N	N	N

Table 1: Summary of IBM Gekko RISC Microprocessor Version 1.1 for DD1.0, 2.0, 2.3, 2.4 Errata

Note: The Gekko errata listed below correspond to the 750-PID8t errata in following manner:
1 = PID8t Erratum #4, 2 = PID8t Erratum #7, 3 = PID8t Erratum #9, 4 = PID8t Erratum #12, 5 = PID8t Erratum #16.

#	Problem	Description	Impact	Solution(s)	Applicable to Version			
					DD 1.0	DD 2.0	DD2.3	DD2.4
21	Dynamic Power Management (DPM)	DPM has been found to cause problems with the floating point unit and with the load/store unit in Gekko mode. Using DPM can cause erroneous results for floating point and load/store operations.	DPM does not offer significant power savings; other power management modes are still available.	DPM should be considered broken on DD1.0 hardware and should never be enabled (that is HID0 bit 11 should always be ‘0’).	Y	Y	N	N
22	Buses float when not driven.	The address and data buses float to an indeterminate state when external pullup resistors are not used.	Unpredictable behaviour may result with external devices that need to see valid logic levels on the data and address buses.	Add pullup resistors to the buses.	Y	N	N	N
23	DD2.0 PVR is wrong.	The PVR value for DD2.0 is same value as DD1.0 (that is, 7000 0100)	The PVR cannot be used to differentiate between DD1.0 and DD2.0.	None	N	Y	N	N
24	I/O Leakage on JTAG Inputs	Some JTAG input receivers leak additional current when held low and operating the I/O’s in high voltage mode.	When held low, each of these pins could have up to 100µA of leakage current.	Operate I/O’s in low voltage mode or tie these inputs high when operating in high voltage mode.	Y	Y	N	N
25	Duplicate Page Table Entries (PTEs) can cause problems.	Duplicate valid entries stored too close to each other have been found to cause problems ranging from incorrect C-bit to a processor hang.	The incorrect C bit may only result in performing a store to virtual memory.Worst case - tablewalker colliding with load/store activity will cause a processor hang.	It is recommended that if identical PTEs are needed, place one in the primary PTEG and the other in the secondary PTEG.	Y	Y	N	N
26	LSU store gather does not work in paired-single mode.	If integer store gathering is enabled (HID0 bit 24 is ‘1’) in paired-single mode, incorrect data is written to memory.	Incorrect data is stored.	Set store gather enable (HID0 bit 24) to zero/off, when paired-single modes are enabled.	Y	Y	N	N
27	Write pipe/$\overline{\text{ARTRY}}$ incompatible	Write gather pipe may hang if write gather requests are ARTYed.	The write pipe state machine may hang.	Do not activate the ARTY pin for write gather requests.	Y	Y	N	N
28	Store data lost during DMA.	A processor load request is received by the data cache while it is busy with a DMA. The load invalidates the line prior to doing the reload, resulting in the store data being lost.	Data may be lost.	Run the data cache in write-thru mode.	Y	Y	N	N
29	GBL level incorrect for table walker loads.	The M bit is not latched correctly resulting in the GBL pin incorrectly driven inactive during the table walker load request.	Coherency will not be maintained for Page Table Entries.	None.	Y	Y	N	N

Table 1: Summary of IBM Gekko RISC Microprocessor Version 1.1 for DD1.0, 2.0, 2.3, 2.4 Errata

Note: The Gekko errata listed below correspond to the 750-PID8t errata in following manner:
1 = PID8t Erratum #4, 2 = PID8t Erratum #7, 3 = PID8t Erratum #9, 4 = PID8t Erratum #12, 5 = PID8t Erratum #16.

#	Problem	Description	Impact	Solution(s)	Applicable to Version			
					DD 1.0	DD 2.0	DD2.3	DD2.4
30	Not able to snoop during nap mode.	Cache logic is unable to transition to run state from nap mode.	Snooping in nap mode may result in a processor hang.	1. Cache coherency must be done under software control. 2. Turn L2 cache off.	Y	Y	N	N
31	DMA/table walker hang.	Processor can hang doing a hardware table walk after a DMA command.	Processor will hang.	None.	Y	Y	Y	N
32	DMA store data incorrect/ ifetch.	If the L2 index for DMA and instruction fetch match, then the BIU state machine allows incorrect data to be stored.	Incorrect DMA store data.	None.	Y	Y	Y	N
33	**stwcx.** after snoop hit may store wrong data.	An instruction sequence that causes the store buffers to fill followed by a **stwcx.** instruction can yield the wrong data stored to memory for the **stwcx.** in the presence of a snoop hit.	An incorrect value will impact succeeding uses of the corresponding semaphore, and may compromise the integrity of that resource.	Place a **sync** instruction before the **stwcx.** - forcing all bus transactions associated with previous store instructions to complete before the **stwcx.** executes.	N	Y	Y	N
34	tablewalk/snoop collision yields false page fault.	Access by tablewalk that hits in data cache leads to a false page fault when snoop push occurs in same cycle.	Incorrect detection of a page fault or incorrect page information.	The page fault handler checks to see if the page being requested is already mapped in memory. If so, the handler can return without modifying the page table, and the retry of the table walk will succeed.	N	Y	Y	N

Erratum #1: L2 cache invalidate may fail 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 stale or incorrect data in the L2 cache which should have been invalidated.

Solution(s)

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 erratum is present in all silicon revisions (software solution is deemed sufficient).

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

Overview

If the target address of a dcbz instruction hits in the Gekko L1 cache at the same time that a snoop is received to that address, the Gekko 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 the Gekko more recent copy of the data), resulting in stale data or a cache coherency violation.

Detailed Description

If the target address of a dcbz instruction hits in the Gekko L1 cache, the Gekko 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, the Gekko 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 the Gekko 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 the Gekko's 3-state cache environment.

Projected Impact

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

Solution(s)

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 erratum is present in all silicon revisions; a fix is not planned at the present time.

Erratum #3: Segment register updates may corrupt data translation

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

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.

Solution(s)

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 erratum is present in all silicon revisions; a fix is not planned at this time.

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.

Solution(s)

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

Status

This is present in all silicon revisions; a fix is not planned at this time.

Erratum #5: Snoop push may occur twice

Overview

When the Gekko snoops a bus transaction and does a snoop push to system memory as a result, it may push the cache line twice. The second push will contain the most recent (most coherent) 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 reallocations in the L1 data cache.
2. In the L2 cache, the cache tag for cache 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 the Gekko snoops and which hits for cache line "A". Internally, the Gekko registers a snoop hit 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 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 causes Gekko instead to incorrectly push the cache line in the L2 castout buffer, and then push the same cache line again (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 Gekko 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 "old" data from memory instead of the "latest" data which is still in the Gekko's snoop push buffer. This may result is data corruption in the system. This type of failure could occur in a system where a common system memory is accessed through two different bus ports, such as through the 60x bus and the PCI bus.

Also, if a new snoop triggers a snoop push from the Gekko while the snoop buffer still contains the previous uncompleted 2nd snoop push, the Gekko 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).

Solution(s)

One work-around is to allow both snoop pushes to be performed before continuing the alternate master that triggered the snoop.

Another work-around is to prevent dirty data from residing in the Gekko's L2 cache. This could be achieved by operating the Gekko's L2 cache in write-through mode by setting the L2WT bit in its L2 configuration register. This solution would work in any system configuration.

Note that in systems where all snoopable memory (i.e., all system memory) is accessed directly and only over 60x bus, the occurrence of a double snoop push would not cause a failure, and a work-around would normally not be required. In this case, the 2 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

Fix implemented in Gekko but limited system level verification completed.

Erratum #6: DMA QCNT Decrements before Load Data is in the Cache

Overview

In some cases, the DMA QCNT field of the HID_G register decrements before the DMA load data is in the cache.

Detailed Description

The problem occurs when the last cache line of a DMA load request misses the data cache. In this case the DMA QCNT is decremented immediately, even if the bus transaction for the DMA load request of the previous cache line is still in progress.

Projected Impact:

There should be no impact, as normally all DMA requests should hit the data cache. However, this problem may be experienced as system code is developed.

Solution(s)

Code all DMA requests to hit the data cache. The DMA machine check interrupt may be used to detect DMA requests which miss the data cache.

Status

Fixed in DD2.0.

Erratum #7: Write Gather is not pipe-lined

Overview

Consecutive bus transactions for the write gather pipe are not pipe-lined

Detailed Description

The write gather implementation only allows one outstanding bus request at a time

Projected Impact

Bus performance may be decreased due to supporting one outstanding bus request at a time.

Solution(s)

None

Status

Fixed in DD2.0.

Erratum #8: Unaligned stfd may cause processor hang

Overview

A store float double (or quantized store) can collide with a tablewalk operation causing a processor hang condition.

Detailed description

The hang condition may occur after a conditional branch instruction when an unaligned float double (or quantized store) occurs that is dependent on a previous operation to the same FPR register. The second access to the unaligned operand must result in a TLB miss. In this case, the processor may hang and a restart is only possible with reset.

Projected Impact

No impact if segmented address translation is not used. If used, the sequence of instructions detailed above will cause a hang condition that may only be cleared with reset.

Solution(s)

Avoid using segmented address translation. If segmented address translation is used, then floating point double operands must be double word-aligned.

Status

Fixed in DD2.0.

Erratum #9: Invalid Transaction Type with DCBZ(M=0) during DMA

Overview

DCBZ with the M bit of the WIMG bits equal to zero may cause an invalid transaction type if the DCBZ causes a copyback wile a DMA transaction is in progress.

Detailed Description

The DMA transaction type may be used for the DCBZ copy back transaction. This can cause bus contention problems if the DMA command is a load.

Projected Impact

Bus contention problems may occur as a result of this error.

Solution(s)

Set up all DCBZ instructions to use an M=1 setting for the WIMG bits.

Status

Fixed in DD2.0.

Erratum #10: DMA Hang due to L2 Request Queue Full

Overview

A DMA load request gets dropped, resulting in a processor hang.

Detailed Description

DMA load requests are run through the L2 request queue, to keep requests in the proper sequence. If the L2 request queue has 2 processor stores pending when the DMA load starts up, there is a potential for the second DMA load to get dropped.

Projected Impact

If this situation occurs, the processor will hang waiting for the second DMA load to complete.

Solution(s)

Clear the L2 request queue with a SYNC instruction before initiating a DMA load, or run with the L2 cache disabled.

Status

Fixed in DD2.0.

Erratum #11: Store Lost due to Queued L/S Requests during DMA

Overview

A processor store gets lost while running DMA because a copy back from the data cache does not occur.

Detailed Description

Load/Store requests which are received while a DMA request is running that miss the cache are queued in the data cache control logic until the DMA completes. The information that gets queued includes the status of the line (the modified bit). The error occurs under the following conditions:

1. The state of the cache line at the time the reload request is made is exclusive.
2. A subsequent store hits the cache line that is going to be replaced by the reload request.

In this case, no copyback will be performed for the line, even though it is now modified due to the store hit.

Projected Impact

Store data will be lost.

Solutions

None at the present time, but to avoid this problem don’t use DMA.

Status

Fixed in DD2.0.

Erratum #12: L2 Reload Data Incorrect during DMA

Overview

Data for an L2 reload due to an instruction fetch request is written incorrectly while DMA is active.

Detailed Description

DMA load requests are run through the L2 request queue. If a DMA load request happens just prior to an instruction fetch request to the L2 which results in a L2 reload, the L2 does not correctly tag the reload request as an I fetch. This causes the reload data to be written incorrectly.

Projected Impact

An incorrect instruction sequence may result if the line that was loaded in the L2 is accessed again.

Solution(s)

Run with the L2 cache disabled

Status

Fixed in DD2.0.

Erratum #13: RESERVE Register Not Updated during DMA

Overview

The RESERVE register used by the LWARX instruction is not updated correctly in some cases.

Detailed Description

If a LWARX instruction is executed while DMA is running, it may get queued until DMA is finished. If the LWARX causes a copy back and the copy back queues are full, then the LWARX request is delayed. The delay results in the RESERVE register not getting updated for the lwarx instruction.

Projected Impact

The LWARX instruction does not function correctly.

Solution

When using DMA, avoid using the LWARX instruction.

Status

Fixed in DD2.0.

Erratum #14: D TLB can be corrupted during DMA

Overview

TLB miss processing by the hardware tablewalker can result in TLB corruption if run concurrently with DMA.

Detailed Description

If a hardware table walker request is received by the cache while a DMA load is running and a DMA store is the next command to be run, then the table walker will try to run while the DMA store is ongoing and the D TLB will be corrupted.

Projected Impact

Paging with DMA running will not operate correctly.

Solution

When using DMA, avoid the usage of the TLB.

Status

Fixed in DD2.0.

Erratum #15: DCBT/DCBTST may cause a hang

Overview

If a DCBT or DCBTST instruction is received while the cache is processing a miss a processor hang condition may result.

Detailed Description

If a DCBT or DCBTST instruction request is received by the cache unit while the cache is processing a miss, the processor will hang if the request is sent from the load/store unit during the cycle that the critical word for the miss is returned.

Projected Impact

The DCBT and DCBTST instructions can not be used to avoid this problem.

Solution

None

Status

Fixed in DD2.0.

Erratum #16: DMA store bus utilization is not optimal

Overview

DMA back to back write performance is not optimal.

Detailed Description

During a sequence of DMA stores, the processor takes 3 bus cycles, versus the 2 expected, in 2:1 mode from the last TA of a DMA store to the TS of subsequent DMA store using the same store queue in the bus interface unit.

Projected Impact

The bus utilization and subsequent processor performance will not be optimal.

Solution

None

Status

Fixed in DD2.0.

Erratum #17: Incorrect load completion during a single step

Overview

The LOAD/STORE unit is incorrectly completing a load in single step mode.

Detailed Description

In single step mode, only one instruction should be completed per step. In this case a store instruction is to be single stepped, with a load instruction immediately following. Both instructions are executed and while the store is waiting to complete, the load completes with the stores' idn (instruction id number to ccc). This results in a possible hang between the LOAD/STORE and the CCC units.

Projected Impact

The single step mode will not function correctly for all cases to enable a smooth debug environment. The exposure for a processor hang or incorrect result exists.

Solution

A sync instruction may be inserted between the store and the load to alleviate this problem during debug.

Status

Fixed in DD2.0.

Erratum #18: Write pipe request gets repeated

Overview

A store request gets queued multiple times in the write gather pipe queue.

Detailed Description

If a write gather pipe request is received while a cache miss is in progress, and the critical word for the cache miss was returned to the load/store unit before the write gather pipe request was made, the request will be queued multiple times in the write gather pipe. This will result in data for the given write gather pipe request getting repeated when the write gather pipe is written to the bus.

Projected Impact

Data integrity problems may occur when using the write gather pipe with the data cache enabled.

Solution

Disable the data cache when using the write gather pipe.

Status

Fixed in DD2.0.

Erratum #19: Store dependency results in stale data

Overview

The LOAD/STORE unit is storing stale data for floating point and quantized stores when there is a store dependency condition.

Detailed Description

Examples of a failing instruction steam:

1. stfd f10, unaligned across dw, tlb miss on both halves
2. fmsub f10,
3. stw
4. fmadd f6,
5. stfd f6, unaligned across dw, dcache hit, tlb hit

The first STFD is an unaligned store with a TLB miss for both stores. The TLB miss provides delay for the succeeding instructions to be dispatched. Once the table walk is complete, the second store of the unaligned operation is executed but it is incorrectly identifying a store dependency condition. The result is to delay the store for one clock cycle which causes the store queue state machine to get out of sync. The error is not seen until the next STFD is executed and a real store dependency is required, waiting for the FMADD to complete. Stale data (F6 data prior to FMADD) is used for the STFD before the FMADD completes writeback to the FPR.

Projected Impact

Unaligned floating point and quantized stores may result with incorrect data written to memory.

Solution

Keep all floating point double precision and quantized stores double word aligned.

or

Keep unaligned FP and PSQ store accesses in the same page, not across different pages.

Status

Fixed in DD2.0.

Erratum #20: DMA load data integrity problem

Overview

In some cases, an extra DMA load request is issued to the BIU. This causes data cache to be out of sync with the BIU.

Detailed Description

If a DMA load request is sent from the data cache to the BIU and another none-DMA cacheable load request is still in the queue fro processing at the time that the load request is received, and extra DMA load request is sent to the BIU from the request queues. This causes the data cache to get out of sync because it gets more data back from the BIU than it has requested.

Projected Impact

Data integrity problems may occur when using DMA loads.

Solution

None

Status

Fixed in DD2.0.

Erratum #21: Dynamic Power Management (DPM)

Overview

DPM has been found to cause problems with the floating point unit and with the load/store unit in Gekko mode. Using DPM can cause erroneous results for floating point and load/store operations.

Detailed Description

This combines Erratum #19 (Issue 20) (L/S & DPM) with the FPU/DPM erratum (NEED NUMBER?) into a general DPM erratum. Essentially, the DPM function does not work with quantized load/stores or paired-single FPU operations for release DD1.0. For DD2.0 and higher the DPM enable/disable bit may be written, but it has no effect on the power consumed (that is, DPM enable does not gate off any clocks) so setting the bit does not affect the quantized load/stores or paired-single FPU operations.

Projected Impact

Small. DPM does not offer significant power savings and other power management modes are still available.

Solution

DPM should be considered broken on DD1.0 hardware and should never be enabled (that is HID0 bit 11 should always be 0). Related to Erratum #19.

Status

Fixed in DD2.3.

Erratum #22: Buses float when not driven.

Overview

The address and data buses float to an indeterminate state when external pullup resistors are not used.

Detailed Description

When the address and data buses are not being driven, the buses float to an unknown level between logical 1 and a logical 0 value. This may cause external devices to react in unpredictable ways.

Projected Impact

Unpredictable behaviour may result with external devices that need to see valid logic levels on the data and address buses.

Solution

Add pullup resistors to the buses.

Status

Fixed in DD2.0.

Erratum #23: DD2.0 PVR is wrong

Overview

The PVR value for DD2.0 is same value as DD1.0 (that is, 7000 0100).

Detailed Description

Both DD2.0 and DD1.0 have the same value (7000 0100).

Projected Impact

The PVR cannot be used to differentiate between DD1.0 and DD2.0.

Solution

None

Status

Version DD2.1 has PVR value of 7000 2201.

Erratum #24: I/O leakage on JTAG inputs

Overview

Some JTAG input receivers leak additional current when held low and I/Os are operated with high voltage (that is, BVSEL = 1, OVDD=2.5V).

Detailed Description

The pullup device on the JTAG pins TCK, L1_TSTCLK, and LSSD_MODE do not completely disable when operating in the 2.5V I/O mode. As a result, additional leakage current occurs when these pins are held low.

Projected Impact

When held low, each of these pins could have no more than 100µa of leakage current.

Solution

Operate I/Os is low voltage mode (that is, BVSEL = 0 and OV_{DD}=1.8V), or when operating in high voltage (that is, BVSEL = 1, OV_{DD}=2.5V), tie the inputs to high.

Status

Fixed in DD2.3.

Erratum #25: Duplicate PTEs can cause problems.

Overview

Duplicate page table entries (PTEs) with only different R and/or C bits are legal in PPC architecture. Duplicate valid entries that are stored too close to each other have been found to cause problems ranging from incorrect C bits to processor hangs.

Detailed Description

This problem occurs when matching PTEs exist in close proximity to each other and have inconsistent C bit values (that is, PTE1 C=0, PTE2 C=1). If PTE1 contains the desired C bit value, and PTE2 arrives too quickly, the hardware tablewalker uses PTE2's C bit value, which results in an incorrect C bit setting.

In the most severe form, this problem can interfere with the load/store unit and cause the load/store state machine to hang.

Projected Impact

In many cases, the incorrect C bit may result in performing a store to virtual memory (hard drive, etc.) that wasn't necessary during paging.

In the worst case, tablewalker colliding with load/store activity causes a processor hang.

Solution

If identical PTEs are needed, place one in the primary PTEG and the other in the secondary PTEG.

Status

Fixed in DD2.3.

Erratum #26: LSU store gather does not work in paired-single mode.

Overview

If integer store gathering is enabled (HID0 bit 24 is 1) in paired-single mode, incorrect data is written to memory.

Detailed Description

When paired-singles mode is enabled then the L/S unit store gathering feature must be disabled by setting the HID0 SGE bit to zero. If HID0 SGE bit is one, incorrect data are written to memory.

Projected Impact

Incorrect data are stored.

Solution

Set store gather enable (HID0 bit 24) to zero, when paired-single modes are enabled.

Status

Fixed in DD2.3.

Erratum #27: Write pipe/$\overline{\text{ARTRY}}$ incompatible.

Overview

The write gather pipe may hang if write gather requests are ARTYed.

Detailed Description

When a write-gather-pipe request is made to the Bus Interface Unit (BIU), the BIU responds with a *grant* signal to indicate that the request has been accepted. If the ARTY pin is activated for this request, the BIU responds with another *grant* signal for the same request. This causes the write pipe state machine to hang if it has another request to go at the time of the second *grant* signal.

Projected Impact

The write pipe state machine may hang.

Solution

Do not activate the ARTY pin for write gather requests.

Status

Fixed in DD2.3.

Erratum #28: Store data lost during DMA

Overview

Store data may be lost during while running DMA commands.

Detailed Description

This error occurs only for very specific instruction timings. A processor load request is received by the data cache while it is busy with a DMA. The load is a miss and a replacement way is selected. After the DMA completes, the data cache starts processing the load. In the same cycle, a processor store occurs which hits the way that the load is going to replace, setting its state to modified. The load then invalidates that line prior to doing the reload, which results in the store data being lost.

Projected Impact

Data integrity problem.

Solution

Run the data cache in write-thru mode.

Status

Fixed in DD2.3.

Erratum #29: GBL pin incorrect for table walker loads

Overview

The GBL pin is not asserted during a hardware table walker load.

Detailed Description

Table walker load requests should force a WIM of 001. The M bit is not latched correctly resulting in the GBL pin incorrectly driven inactive during the table walker load request.

Projected Impact

Coherency will not be maintained for Page Table Entries.

Solution

None.

Status

Fixed in DD2.3.

Erratum #30: Snoop results in processor hang during nap mode.

Overview

Hang condition results when snooping in nap mode.

Detailed Description

When the processor is in nap mode, a hang condition will result if a snoop hit to a modified line occurs in the L2 cache. When the processor comes out of nap mode, it will continue to run until a miss in both the L1 and L2 caches requires a bus cycle to main memory. At this point, the processor will stall waiting for data it will never get. The snoop writeback is also not performed.

Projected Impact

Snooping in nap mode may result in a processor hang.

Solution

1. Cache coherency must be done under software control.
2. Turn L2 cache off.

Status

Fixed in DD2.3.

Erratum #31: DMA/table walker hang

Overview

Processor can hang doing a hardware table walk after a DMA command.

Detailed Description

The following sequence of events will hang the processor.

A DMA command, followed by a single beat bus operation by the processor. This is followed by a processor operation which causes a table walk. If the matching page table entry is the last one of the Page Table Entry Group (PTEG), the processor will hang.

Projected Impact

Processor will hang.

Solution

None.

Status

Fix is planned for DD2.4.

Erratum #32: DMA store data incorrect/ifetch

Overview

DMA store is incorrect.

Detailed Description

The following sequence of events will cause incorrect store data for a DMA store command.

A DMA store command is in progress. The address of the DMA store has been sent to the BIU, but the data has not been sent because the DMA stores are pipelined. The processor makes an instruction fetch request to the L2, which misses in the L2 and causes a reload. The L2 index of the DMA store matches the index of the instruction fetch which is being reloaded. This stalls the DMA store, which prevents the data for the DMA store from being sent to the BIU. After the instruction fetch bus request completes, the DMA store goes out on the bus before the data has been sent to the BIU, resulting in incorrect data.

Projected Impact

Incorrect DMA store data.

Solution

None.

Status

Fix is planned for DD2.4.

Erratum #33: stwcx. after snoop hit may store wrong data

Overview

An instruction sequence that causes the store buffers to fill followed by a store word condtional (**stwcx**.) instruction can yield the wrong data stored to memory for the **stwcx.** in the presence of a snoop hit.

Detailed Description

Conditons which precipitate a failure are are follows.

1. The 2 L1 store queues are full (stores for CI, WT, CB).
    a. an unaligned store requiring two bus cycles (i.e., **stfd**)
    b. two separate single beat stores
    c. L1 line replacement copybacks
2. The first store has started its address tenure on the 60x bus.
3. A snoop cycle is detected and hits modified data in the L1 cache resulting in a snoop copyback.
4. A **stwcx.** with a prior **lwarx** setting a reservation is waiting for the first store to finish on the bus and freeing up an entry in the store queue.
5. When the store queue has an entry, the **stwcx.** collides with the snoop copyback loading the snoop queue. Invalid data gets loaded into the L1 store queue for the **stwcx**. The snoop queue gets its correct data.
6. The snoop copyback is performed on the bus, then the second store, and finally the **stwcx.** with incorrect data.

Generalized Code Sequence:

1. Any instruction sequence resulting in the store queue (2 entries) full. The following instructions will load one or more entries in the store queue.
    - Cache inhibited or write through, aligned or unaligned stores.
    - Cacheable loads or stores resulting in an L1 line replacement copyback.
    - Cache-ops (**dcbz**,**dcbf**,**dcbst**,**dcbt**) resulting in an L1 copyback.
    - Cacheable lwarx resulting in an L1 line replacement copyback.

    Bus wait states and snooping activity will directly affect the store queue state.
2. A **stwcx.** instruction with prior reservation set.

Projected Impact

The data stored by a **stwcx.** is often a counter or an identifier used to control access to a data structure or other resource. An incorrect value will have its impact on succeeding uses of the corresponding semaphore, and may compromise the integrity of that resource.

Solution

Place a **sync** instruction before the **stwcx**. instruction to prevent the problem from occurring, by forcing all bus transactions associated with previous store instructions to complete first before the **stwcx**. executes.

Status

A fix is planned for DD2.4.

Erratum #34: Tablewalk/snoop collision yields false page fault.

Overview

An access by the tablewalk hardware that hits in the data cache leads to a spurious page fault if a snoop push occurs on the same cycle.

Detailed Description

The conditions for this fail to occur are as follows.

1. A tablewalk is in progress, due to a TLB miss.
2. The cache line containing the page table entry (PTE) being searched for is in the L1 cache.
3. The PTE being searched for is the first entry in the cache line.
4. A snoop on the bus hits a modified line, causing a push.
5. The timing of the tablewalk procedure and the snoop is such that both attempt to access the data cache on the same cycle.

The outcome of these conditions being met is that the tablewalker gets the wrong data for the first PTE in the cache line. The snoop push data is written correctly to the bus.

Projected Impact

If the PTE that is read incorrectly by the tablewalker is the one being searched for, the tablewalk search will fail, leading to an incorrect detection of a page fault. The impact will be a function of the system-level page fault handler. The other scenario is that the wrong data in the first PTE of the cache line matches the sought PTE causing the tablewalk operation to provide incorrect page information to the processor.

Solution

The page fault handler can reject spurious page faults by checking to see if the page being requested is already mapped in memory. If so, the handler can return without modifying the page table, and the ensuing retry of the table walk will succeed.

Status

A fix is planned for DD2.4.